The Design and Implementation of a Two and Three-Dimensional Triangular Irregular Network based GIS

Volume II

Alias Abdul-Rahman

July, 2000

A thesis submitted for the degree of Doctor of Philosophy

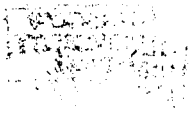
UNIVERSITY
of
GLASGOW

Department of Geography & Topographic Science
University of Glasgow

Volume II

Software Code

This is the Volume II of this thesis. It lists all the C++ codes for all the software developed in this research using Borland C++ version 5.02 compiler.



2D Rasterization Code

The following gives the complete code for the 2D rasterization program (called 2D RASTER).

```
 //*************************************************************************////
//      A Program to rasterize points in 2D                                    //
//                Input: XY world coordinates                                  //
//                Output: MPD and MPI ILWIS format                             //
//                                                                             //
    Copyright (c) Alias Abdul-Rahman, 1999
// ************************************************************************//

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <io.h>
#include <conio.h>
#include <iomanip.h>
#include <string>

const int rowsize = 800;
const int colsize = 800;

typedef struct PointStruct                    // structure for XY world
coordinates
  {
   double x;
   double y;
   double z;
  } PointType;

PointType Point;

typedef short Image[rowsize][colsize];    // Image: Pixel variable

// the MPI file structure
typedef struct MpiStruct
          {
          short Nscanlines;       // no. of image rows
          short Npixels;          // no. of image columns
          short Pvmin;            // pixel's minimum value
          short Pvmax;            // pixel's maximum value
          short Maptype;          // map type
          short Patch;            // patch type
          short Scale;            // scale factor
          short Crdtype;          // coordinate type
          float a11;              // transformation coefficient
          float a12;              //       "
          float a21;              //       "
          float a22;              //       "
          float b1;               //       "
```

```
          float b2;              //       "
          } MpiType;

MpiType MPI;

Image Pixel;

int xlength, ylength;
int pixsize;
short maxrow, maxcol;
short npnt, pnt, maxpnt;
float scalex, scaley, scale;
double xmin, ymin, xmax, ymax;
FILE* MPDfile;
FILE* MPIfile;


// prototype section
void GetXYZfile(double&, double&, double&, double&);
void WriteMPIfile();
void GetPixelSize(int&);
void GetNumRowCol(int, int&, int&);
void MakeRaster(PointType, int, int, int);
void DisplayPixel();
void PressAnyKey();

// Main
void main()
  {
   int i;

   GetXYZfile(xmin, ymin, xmax, ymax);
   GetPixelSize(pixsize);
   GetNumRowCol(pixsize, xlength, ylength);
   MakeRaster(Point, pixsize, xlength, ylength);
   WriteMPIfile();
   DisplayPixel();
   PressAnyKey();

  }

// definition section
void GetXYZfile(double& xmin, double& ymin, double& xmax, double& ymax)
  {
   string charX, charY, charZ;

   ifstream XYZfile("c:\\data\\lochgrd.xyz");

     if (! XYZfile)
       {
        cerr << "Point file not found" << endl;
        exit(1);
       }

   // reads the file header
   XYZfile >> charX >> charY >> charZ;

   npnt = 1;
   XYZfile >> Point.x >> Point.y >> Point.z;
```

```
    // to calculate min xy, max xy
    xmin = Point.x;
    ymin = Point.y;
    xmax = Point.x;
    ymax = Point.y;

    do
      {
       printf("\r");
       cout << "Reading points ... " << npnt;

       XYZfile >> Point.x >> Point.y >> Point.z;
         if (xmin > Point.x)
             xmin = Point.x;

         if (ymin > Point.y)
             ymin = Point.y;

         if (xmax < Point.x)
             xmax = Point.x;

         if (ymax < Point.y)
             ymax = Point.y;

      npnt ++;
      } while (! XYZfile.eof());

    xlength = xmax - xmin;
    ylength = ymax - ymin;
    maxpnt = npnt - 1;
    XYZfile.close();

  }

void GetPixelSize(int& pixsize)
  {
   cout << endl;
   cout << "Enter pixel size (in meter): " << endl;
   cin >> pixsize;
  }

void GetNumRowCol(int pixsize, int& xlength, int& ylength)
  {
   xlength = xmax - xmin;
   ylength = ymax - ymin;
   maxrow = int (ylength / pixsize) + 1;
   maxcol = ceil(int (xlength / pixsize)) + 1;
  }

void MakeRaster(PointType Point, int pixsize, int xlength, int ylength)
  {
   int i, r, c, j;
   short PointID;
   short row, col;
   long cursor;
   float scale;
   string charX, charY, charZ;

   scale = 1.0 / float(pixsize);
```

```
cout << endl;
cout << "xmin, ymin: " << xmin << ", " << ymin << endl;
cout << "xmax, ymax: " << xmax << ", " << ymax << endl;
cout << "pix size : " << pixsize << endl;
cout << "xlength : " << xlength << endl;
cout << "ylength : " << ylength << endl;
cout << "xmin :" << xmin << endl;
cout << "ymin :" << ymin << endl;
cout << "maxrow :" << maxrow << endl;
cout << "maxcol :" << maxcol << endl;

cout << setiosflags(ios::showpoint) << endl;
cout << setprecision(8) << "scale   : " << scale << endl;

ifstream XYZfile("c:\\data\\lochgrd.xyz");

// reads the file header
XYZfile >> charX >> charY >> charZ;

if (! XYZfile)
    {
     cerr << "Point file not found" << endl;
     exit(1);
    }

FILE* MPDfile;
char mpdfilename[] = "c:\\data\\lochgrd.MPD";
MPDfile = fopen(mpdfilename, "wb");

pnt = 1;

while (! XYZfile.eof())
  {
   XYZfile >> Point.x >> Point.y >> Point.z;
   col = int(scale * (Point.x - xmin)) + 1;
   row = int((-1 * scale) * (Point.y - ymin)) + maxrow + 1;
   if (XYZfile.eof() == true)
     pnt = pnt - 1;
   PointID = pnt;
   Pixel[row][col] = PointID;
   printf("\r");
   cout << "Rasterizing the points ... " << pnt;
   pnt ++;
  }

pnt = pnt-1;
maxpnt = pnt;

for (row = 1; row <= maxrow+1; row ++)
  {
   fwrite(Pixel[row], sizeof(short)* maxcol, 1, MPDfile);
  }

cout << endl;
cout << endl;
cout << " Total points : " << maxpnt;

XYZfile.close();
fclose(MPDfile);

}
```

```
void WriteMPIfile()
  {
   FILE* MPIfile;
   char mpifilename[] = "c:\\data\\lochgrd.MPI";
   MPIfile = fopen(mpifilename, "wb");

   // assign the values for the MPI file
   MPI.Nscanlines = maxrow + 1;
   MPI.Npixels = maxcol;
   MPI.Pvmin = 1;
   MPI.Pvmax = maxpnt;
   MPI.Maptype = 2;
   MPI.Patch = 0;
   MPI.Scale = 0;
   MPI.Crdtype = 1;
   MPI.a11 = scale;
   MPI.a12 = 0;
   MPI.a21 = 0;
   MPI.a22 = -scale;
   MPI.b1 = 1;
   MPI.b2 = maxrow + 1;

   // write out the above values in the MPI file
   fwrite(&MPI, sizeof(MpiType), 1, MPIfile);
   fclose(MPIfile);
  }


void DisplayPixel()
  {
   int i, j;
   cout << endl;
   cout << endl;
   cout << "Displaying pixel value ..." << endl;
   cout << endl;

   for (i = 0; i <= maxrow; i ++)
     {
      //for (j = 0; j <= maxcol; j ++)
        //{
         printf("\r");
         cout << "[row]: " << i;

         //cout << "Pixel[row][col]: Pixel[" << i << "][" << j <<
                 //"]: " << Pixel[i][j];

        //}
     }
  }
```

```
void PressAnyKey()
  {
   cout << endl;
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }
```

Constrained 2D Rasterization Code

The following gives the complete code for the constrained 2D rasterization program.

```
//*************************************************************************///
//      A Program to rasterize points and edges or arcs in 2D              //
//                Input: XYZ coordinates                                   //
//                Output: MPD and MPI ILWIS format                         //
//                                                                         //
    Copyright (c) Alias Abdul-Rahman, 1999
// //*************************************************************************/

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <io.h>
#include <conio.h>
#include <iomanip.h>
#include <string>

const int rowsize = 800;
const int colsize = 800;
const int maxarc = 200;
const int maxpoint = 1000;

typedef struct PointStruct          // structure for XY world coordinates
  {
   float x;
   float y;
   float z;
  } PointType;
PointType* Point[maxpoint];

typedef struct ArcStruct            // structure for Arc table
  {
   int StartNode;
   int EndNode;
  } ArcType;
ArcType* Arc[maxarc];

typedef short Image[rowsize][colsize];    // Image: Pixel variable

// the MPI file structure
typedef struct MpiStruct
          {
          short Nscanlines;       // no. of image rows
          short Npixels;          // no. of image columns
          short Pvmin;            // pixel's minimum value
          short Pvmax;            // pixel's maximum value
          short Maptype;          // map type
```

```
          short Patch;          // patch type
          short Scale;          // scale factor
          short Crdtype;        // coordinate type
          float a11;            // transformation coefficient
          float a12;            //       "
          float a21;            //       "
          float a22;            //       "
          float b1;             //       "
          float b2;             //       "
          } MpiType;

MpiType MPI;
Image Pixel;

int numarc;
float xlength, ylength;
int pixsize;
short maxrow, maxcol;
short npnt, pnt, maxpnt;
float scalex, scaley;
float scale;
float xmin, ymin, xmax, ymax;
FILE* MPDfile;
FILE* MPIfile;


// prototype section
void GetXYZfile();
void GetXYMinMax(float&, float&, float&, float&);
void GetArcFile();
void GetMiddleXY(float, float, float, float, float&, float&);
void GetXYZforArcNodes(int, float&, float&, float&, float&,
                       short&, short&);
void WriteMPIfile();
void GetPixelSize(int&);
void GetNumRowCol(int, float&, float&);
void MakeRaster(int, float, float);
void WriteToFile(Image Pixel);
void DisplayPixel();
void PressAnyKey();
void AllocateMemory();
void DeallocateMemory();

// Main
void main()
  {
   int i;
   AllocateMemory();
   GetXYZfile();
   GetXYMinMax(xmin, ymin, xmax, ymax);
   GetPixelSize(pixsize);
   GetNumRowCol(pixsize, xlength, ylength);
   MakeRaster(pixsize, xlength, ylength);
   WriteMPIfile();
   DisplayPixel();
   DeallocateMemory();
   PressAnyKey();
  }

// Implementation section
```

```
void AllocateMemory()
  {
   for (int i = 0; i < maxpoint; i ++)
     Point[i] = new PointType;

   for (int ii = 0; ii < maxarc; ii ++)
     Arc[ii] = new ArcType;
  }

void DeallocateMemory()
  {
   for (int i = 0; i < maxpoint; i ++)
     delete [] Point[i];

   for (int ii = 0; ii < maxarc; ii ++)
     delete [] Arc[ii];
  }

void GetXYZfile()
  {
   string charX, charY, charZ;

   //ifstream XYZfile("c:\\data\\lochass.xyz");
   //ifstream XYZfile("c:\\data\\small.xyz");
   ifstream XYZfile("c:\\data\\p2.xyz");

     if (! XYZfile)
       {
        cerr << "Point file not found" << endl;
        exit(1);
       }

   // reads the file header
   XYZfile >> charX >> charY >> charZ;

   int npnt = 1;
   while (! XYZfile.eof())
     {
      XYZfile >> Point[npnt]->x >> Point[npnt]->y >> Point[npnt]->z;
      printf("\r");
      cout << "Reading points ... " << (npnt-1);
      npnt ++;
     }

  }
void GetXYMinMax(float& xmin, float& ymin, float& xmax, float& ymax)
  {
   string charX, charY, charZ;
   float x, y, z;

   //ifstream XYZfile("c:\\data\\lochass.xyz");
   //ifstream XYZfile("c:\\data\\small.xyz");
   ifstream XYZfile("c:\\data\\p2.xyz");

     if (! XYZfile)
       {
        cerr << "Point file not found" << endl;
        exit(1);
       }
```

```
   // reads the file header
   XYZfile >> charX >> charY >> charZ;

   npnt = 1;
   XYZfile >> x >> y >> z;

   // to calculate min xy, max xy
   xmin = x;
   ymin = y;
   xmax = x;
   ymax = y;

   while (! XYZfile.eof())
     {
      //npnt ++;
      XYZfile >> x >> y >> z;

        if (xmin > x)
            xmin = x;

        if (ymin > y)
            ymin = y;

        if (xmax < x)
            xmax = x;

        if (ymax < y)
            ymax = y;

      npnt ++;
     }

   xlength = xmax - xmin;
   ylength = ymax - ymin;
   maxpnt = npnt - 1;
   XYZfile.close();

  }

void GetArcFile(int& numarc)
  {
   string charsnode, charenode;
   //ifstream ARCfile("c:\\data\\lake.arc");
   //ifstream ARCfile("c:\\data\\small.arc");
   ifstream ARCfile("c:\\data\\p2.arc");

     if (! ARCfile)
       {
        cerr << "Arc file not found" << endl;
        exit(1);
       }

   // reads the file header
   ARCfile >> charsnode >> charenode;

   int narc = 0;
   while (! ARCfile.eof())
     {
      narc ++;
      printf("\r");
      cout << "Reading arcs ... " << narc;
```

```
      ARCfile >> Arc[narc]->StartNode >> Arc[narc]->EndNode;
     }
   numarc = narc-1;

  }

void GetMiddleXY(float x1, float y1, float x2, float y2, float& xm, float& ym)
  {
   xm = (x1 + x2) / 2;
   ym = (y1 + y2) / 2;
  }

void GetXYZforArcNodes(int i, float& xs, float& ys, float& xe, float& ye,
                       short& sNd, short& eNd)
  {
   sNd = Arc[i]->StartNode;
   eNd = Arc[i]->EndNode;
   xs = Point[sNd]->x;
   ys = Point[sNd]->y;
   xe = Point[eNd]->x;
   ye = Point[eNd]->y;
  }

void GetPixelSize(int& pixsize)
  {
   cout << endl;
   cout << "Enter pixel size (in meter): " << endl;
   cin >> pixsize;
  }

void GetNumRowCol(int pixsize, float& xlength, float& ylength)
  {
   xlength = xmax - xmin;
   ylength = ymax - ymin;
   maxrow = int (ylength / pixsize) + 1;
   maxcol = ceil(int (xlength / pixsize)) + 1;
  }

void MakeRaster(int pixsize, float xlength, float ylength)
  {
   int r, c, j;
   short PointID;
   short row, col;
   double scale;
   string charX, charY, charZ;
   string charsnode, charenode;

   int narc;
   float xstart, ystart;
   float xend, yend;
   float xmid, ymid;
   short sNd, eNd;
   int numarc;

   int i;
   float dx, dy;
   float dx2, dy2;
   float x, y;
   float temp;
   float p;
```

```
float m;
bool swap;

scale = 1.0 / pixsize;

cout << endl;
cout << "xmin, ymin: " << xmin << ", " << ymin << endl;
cout << "xmax, ymax: " << xmax << ", " << ymax << endl;
cout << "pix size : " << pixsize << endl;
cout << "xlength : " << xlength << endl;
cout << "ylength : " << ylength << endl;
cout << "xmin :" << xmin << endl;
cout << "ymin :" << ymin << endl;
cout << "maxrow :" << maxrow << endl;
cout << "maxcol :" << maxcol << endl;

cout << setiosflags(ios::showpoint) << endl;
cout << setprecision(8) << "scale   : " << scale << endl;

//ifstream XYZfile("c:\\data\\lochass.xyz");
//ifstream XYZfile("c:\\data\\small.xyz");
ifstream XYZfile("c:\\data\\p2.xyz");

// reads the file header
XYZfile >> charX >> charY >> charZ;

if (! XYZfile)
    {
     cerr << "Point file not found" << endl;
     exit(1);
    }

FILE* MPDfile;
char mpdfilename[] = "c:\\data\\p2.MPD";
MPDfile = fopen(mpdfilename, "wb");

pnt = 1;

while (! XYZfile.eof())
  {
   XYZfile >> Point[npnt]->x >> Point[npnt]->y >> Point[npnt]->z;
   col = int(scale * (Point[npnt]->x - xmin)) + 1;
   row = int((-1 * scale) * (Point[npnt]->y - ymin)) + maxrow + 1;
   if (XYZfile.eof() == true)
     pnt = pnt - 1;
   PointID = pnt;
   Pixel[row][col] = PointID;
   printf("\r");
   cout << "Rasterizing the points ... " << pnt;
   pnt ++;
  }

pnt = pnt-1;
maxpnt = pnt;

GetArcFile(numarc);

for (int t = 1; t <= numarc; t ++)
  {
   GetXYZforArcNodes(t, xstart, ystart, xend, yend, sNd, eNd);
   GetMiddleXY(xstart, ystart, xend, yend, xmid, ymid);
```

```
m = (yend - ystart) / (xend - xstart);

if ((m > 0) && (m < 1))
  {
   swap = false;
   if (xstart > xend)
     {
      temp = xend;
      xend = xstart;
      xstart = temp;

      temp = yend;
      yend = ystart;
      ystart = temp;
      swap = true;
     }

   dx = xend - xstart;
   dy = yend - ystart;

   dx2 = 2 * dx;
   dy2 = 2 * dy;

   col = int(scale * (xstart - xmin)) + 1;
   row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
   if (swap == true)
     {
      if (xstart > xmid)
         PointID = sNd;
      else
         PointID = eNd;
      Pixel[row][col] = PointID;
     }
   else
     {
      if (xstart < xmid)
         PointID = sNd;
      else
         PointID = eNd;
      Pixel[row][col] = PointID;
     }
   // initialise the error
   p = (dy2) - dx;
   xstart ++;

   while (xstart < xend)
      {
       if (p > 0)
         {
          p = p + dy2 - dx2;
          ystart ++;
         }
       else
         {
          p = p + dy2;
         }

       col = int(scale * (xstart - xmin)) + 1;
       row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
       if (swap == true)
         {
```

```
            if (xstart > xmid)
               PointID = sNd;
            else
               PointID = eNd;
            Pixel[row][col] = PointID;
           }
         else
           {
            if (xstart < xmid)
               PointID = eNd;
            else
               PointID = sNd;
            Pixel[row][col] = PointID;
           }
         xstart ++;
        }
    }

// to handle case slope > 1
if (m > 1)
  {
   swap = false;
   if (ystart > yend)
     {
      temp = xend;
      xend = xstart;
      xstart = temp;

      temp = yend;
      yend = ystart;
      ystart = temp;
      swap = true;
     }

   dx = xend - xstart;
   dy = yend - ystart;

   dx2 = 2 * dx;
   dy2 = 2 * dy;

   col = int(scale * (xstart - xmin)) + 1;
   row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
   if (swap == true)
     {
      if (ystart > ymid)
         PointID = sNd;
      else
         PointID = eNd;
      Pixel[row][col] = PointID;
     }
   else
     {
      if (ystart < ymid)
         PointID = sNd;
      else
         PointID = eNd;
      Pixel[row][col] = PointID;
     }
   // initialise the error
   p = (dx2) - dy;
```

```
    ystart ++;

    while (ystart < yend)
       {
        if (p > 0)
          {
           p = p + dx2 - dy2;
           xstart ++;
          }
        else
          {
           p = p + dx2;
          }

        col = int(scale * (xstart - xmin)) + 1;
        row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
        if (swap == true)
          {
           if (ystart > ymid)
              PointID = sNd;
           else
              PointID = eNd;
           Pixel[row][col] = PointID;
          }
        else
          {
           if (ystart > ymid)  // was <
              PointID = sNd;
           else
              PointID = eNd;
           Pixel[row][col] = PointID;
          }
        ystart ++;
       }
  } // end of case slope > 1

// to handle slope m < 0 && m > -1
if ((m < 0) && (m >= -1))
  {
    swap = false;
    if (xstart > xend)
      {
       temp = xend;
       xend = xstart;
       xstart = temp;

       temp = yend;
       yend = ystart;
       ystart = temp;
       swap = true;
      }

    dx = xend - xstart;
    dy = yend - ystart;

    dx2 = 2 * dx;
    dy2 = 2 * dy;

    col = int(scale * (xstart - xmin)) + 1;
    row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
```

```
    if (swap == true)
      {
       if (xstart > xmid)
         PointID = sNd;
       else
         PointID = eNd;
       Pixel[row][col] = PointID;
      }
    else
      {
       if (xstart < xmid)
          PointID = sNd;
       else
          PointID = eNd;

       Pixel[row][col] = PointID;
      }
    // initialise the error
    p = (dx2) + dy;
    xstart ++;

    while (xstart < xend)
       {
        if (p > 0)
          {
           p = p - (dy2 + dx2);
           ystart --;
          }
        else
          {
           p = p - dy2;
          }

        col = int(scale * (xstart - xmin)) + 1;
        row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
        if (swap == true)
          {
           if (xstart > xmid)
              PointID = sNd;
           else
              PointID = eNd;
           Pixel[row][col] = PointID;
          }
        else
          {
           if (xstart < xmid)
              PointID = sNd;
           else
              PointID = eNd;

           Pixel[row][col] = PointID;
           }

        xstart ++;
       }

   }  // end m > -1 && m < 0

// to handle slope m < -1
if (m < -1)
```

```
{
 swap = false;
 if (ystart > yend)
   {
    temp = xend;
    xend = xstart;
    xstart = temp;

    temp = yend;
    yend = ystart;
    ystart = temp;
    swap = true;
   }

 dx = xend - xstart;
 dy = yend - ystart;

 dx2 = 2 * dx;
 dy2 = 2 * dy;

 col = int(scale * (xstart - xmin)) + 1;
 row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
 if (swap == true)
   {
    if (ystart > ymid)
      PointID = sNd;
    else
      PointID = eNd;
    Pixel[row][col] = PointID;
   }
 else
   {
    if (ystart < ymid)
       PointID = sNd;
    else
       PointID = eNd;
    Pixel[row][col] = PointID;
   }

 // initialise the error
 p = dx2 + dy;
 ystart ++;

 while (ystart < yend)
    {
     if (p > 0)
       {
        p = p - (dx2 + dy2);
        xstart --;
       }
     else
       {
        p = p - dx2;
       }

     col = int(scale * (xstart - xmin)) + 1;
     row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
     if (swap == true)
       {
        if (ystart > ymid)
```

```
                PointID = sNd;
              else
                PointID = eNd;
              Pixel[row][col] = PointID;
            }
          else
            {
             if (ystart < ymid)
                PointID = sNd;
             else
                PointID = eNd;
             Pixel[row][col] = PointID;
            }
          ystart ++;
         }
    }  // end m < -1

// handle horizontal line
if (ystart == yend)
    {
     while (xstart > xend)
       {
        col = int(scale * (xstart - xmin)) + 1;
        row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
        if (xstart > xmid)
           PointID = sNd;
        else
           PointID = eNd;

        Pixel[row][col] = PointID;
        xstart = xstart - 1;
       }

        while (xstart < xend)
          {
           col = int(scale * (xstart - xmin)) + 1;
           row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
           if (xstart < xmid)
              PointID = sNd;
           else
              PointID = eNd;

           Pixel[row][col] = PointID;
           xstart = xstart + 1;
          }

    }

// handle vertical line
if (xstart == xend)
    {
     while (ystart < yend)
       {
        col = int(scale * (xstart - xmin)) + 1;
        row = int((-1 * scale) * (ystart - ymin)) + maxrow + 1;
        if (ystart < ymid)
          PointID = sNd;
        else
          PointID = eNd;
```

```
            Pixel[row][col] = PointID;
            ystart = ystart + 1;
           }
        }
     } // the t loop


   // write to file
   for (row = 1; row <= maxrow+1; row ++)
     {
      fwrite(Pixel[row], sizeof(short)* maxcol, 1, MPDfile);
     }

   cout << endl;
   cout << endl;
   cout << " Total points : " << maxpnt;

   XYZfile.close();
   fclose(MPDfile);
  }

void WriteMPIfile()
  {
   FILE* MPIfile;
   char mpifilename[] = "c:\\data\\p2.MPI";
   MPIfile = fopen(mpifilename, "wb");

   // assign the values for the MPI file
   MPI.Nscanlines = maxrow + 1;
   MPI.Npixels = maxcol;
   MPI.Pvmin = 1;
   MPI.Pvmax = maxpnt;
   MPI.Maptype = 2;
   MPI.Patch = 0;
   MPI.Scale = 0;
   MPI.Crdtype = 1;
   MPI.a11 = scale;
   MPI.a12 = 0;
   MPI.a21 = 0;
   MPI.a22 = -scale;
   MPI.b1 = 1;
   MPI.b2 = maxrow + 1;

   // write out the above values in the MPI file
   fwrite(&MPI, sizeof(MpiType), 1, MPIfile);
   fclose(MPIfile);
  }


void DisplayPixel()
  {
   int i, j;
   cout << endl;
   cout << endl;
   //cout << "Displaying pixel value ..." << endl;
   cout << endl;

   for (i = 0; i <= maxrow; i ++)
     {
      //for (j = 0; j <= maxcol; j ++)
        //{
```

```
          printf("\r");
          cout << "Rasterizing row .... " << i;
          //cout << "Pixel[row][col]: Pixel[" << i << "][" << j <<
                  //"]: " << Pixel[i][j];

         //}
      }
  }

void PressAnyKey()
  {
   cout << endl;
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }
```

2D Distance Transformation (DT)

The following gives the complete codes for the 2D Distance Transformation (DT).

```
//*************************************************************//
//                                                             //
// An Object-Oriented Program: Distance Transformation (DT)    //
//  Input: MPD and MPI file (rasterised points files)          //
//  Output: MPD and MPI file(DT raster files)                  //
//                                                             //
// Copyright (c) Alias Abdul-Rahman, 1999                      //
//*************************************************************//

// File: TDistanceTransform.h

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <io.h>
#include <iomanip.h>
#include <malloc.h>
#include <conio.h>
#include <limits.h>
#include <stddef.h>

#define masksize 8

class TDistanceTransform
  {
   public:
        // member data
        typedef struct MpiStruct
           {
            short Nscanlines;
            short Npixels;
            short Pvmin;
            short Pvmax;
            short Maptype;
            short Patch;
            short Scale;
            short Crdtype;
            float a11;
            float a12;
            float a21;
            float a22;
            float b1;
            float b2;
           } MpiType;

        typedef short Image;
```

```
        typedef Image* ImagePtr;
        typedef ImagePtr* ImagePPtr;
        typedef short Mask[9];

        MpiType MPI, MpiOut;
        Mask MaskPix;
        int row;
        int col;
        int r;
        ImagePPtr Pixel;
        int maxrow;
        int maxcol;
        FILE* MPIfile;
        FILE* MPDfile;
        FILE* MPIofile;
        FILE* MPDofile;

        // member functions
        void GetMPIInputFile();
        void GetMPDInputFile();
        void ReadImage(ImagePPtr& Pixel);
        void PrintPixel(ImagePPtr Pixel);
        void WriteMPIOutputFile();
        void WriteMPDOutputFile();
        void SetBackground(ImagePPtr Pixel, int Bg, int Fg);
        void GetUpperMask(int r, int c, ImagePPtr Pixel, Mask& MaskPix);
        void GetLowerMask(int r, int c, ImagePPtr Pixel, Mask& MaskPix);
        int MinByIndex(int from, int to);
        int Min5(int a, int b, int c, int d, int e);
        void DistancePassOne(ImagePPtr Pixel);
        void DistancePassTwo(ImagePPtr Pixel);
        void ForwardDistance();
        void BackwardDistance();
        void PressAnyKey();
        void Title();
        TDistanceTransform();          // constructor
        ~TDistanceTransform();         // destructor

};
```

```
//*************************************************************//
//                                                             //
// An Object-Oriented Program: Distance Transformation (DT)    //
//  Input: MPD and MPI file (rasterised points files)          //
//  Output: MPD and MPI file(DT raster files)                  //
// Copyright (c)Alias Abdul-Rahman, 1999                       //
//*************************************************************//

// File: TDistanceTransform.cpp

#include "TDistanceTransform.h"


void main()
  {
   TDistanceTransform DT;        // DT is an object
   DT.Title();
   TDistanceTransform();         // Allocate memory
   DT.GetMPDInputFile();
   DT.ForwardDistance();
   DT.BackwardDistance();
   DT.WriteMPDOutputFile();
   DT.WriteMPIOutputFile();
   DT.PressAnyKey();
  } // end of main


// Definitions Section

TDistanceTransform :: TDistanceTransform()          // constructor
  {
   // Allocate memory
   GetMPIInputFile();
   Pixel = new ImagePtr[maxrow];
   for (int r = 0; r < maxrow; r ++)
     {
      Pixel[r] = new Image[maxcol];
     }
  }

TDistanceTransform :: ~TDistanceTransform()         // destructor
  {
   // Deallocate memory
   for (int r = 0; r < maxrow ; r ++)
     {
      delete [] Pixel;
     }
   delete [] Pixel;
  }

void TDistanceTransform :: GetMPIInputFile()
  {
   //FILE* MPIfile;

   char mpifilename[] = "c:\\data\\pentx.mpi";
   MPIfile = fopen(mpifilename, "rb");

   if (MPIfile == NULL)
     {
       cerr << " Could not open MPI file !" << endl;
```

```
        exit(1);
      }
    else
      {
       cout << "Information of  " << mpifilename << endl;
       fread(&MPI, sizeof(MpiType), 1, MPIfile);
       int size = sizeof(MpiType);

       maxrow = MPI.Nscanlines;
       maxcol = MPI.Npixels;

       cout << "MPI file size       : " << size << endl;
       cout << endl;
       cout << "Number of lines     : " << MPI.Nscanlines << endl;
       cout << "Number of columns   : " << MPI.Npixels << endl;
       cout << "Minimum value       : " << MPI.Pvmin << endl;
       cout << "Maximum value       : " << MPI.Pvmax << endl;
       cout << "Map type            : " << MPI.Maptype << endl;
       cout << "Patched             : " << MPI.Patch << endl;
       cout << "Scale (power of 10) : " << MPI.Scale << endl;
       cout << "Coordinate type     : " << MPI.Crdtype << endl;
       cout << "Transformation parameters :" << endl;
       cout << " a11  : " << MPI.a11 << endl;
       cout << " a12  : " << MPI.a12 << endl;
       cout << " a21  : " << MPI.a21 << endl;
       cout << " a22  : " << MPI.a22 << endl;
       cout << " b1   : " << MPI.b1 << endl;
       cout << " b2   : " << MPI.b2 << endl;
       fclose(MPIfile);
      }
  }

void TDistanceTransform :: GetMPDInputFile()
  {
   char mpdifilename [] = "c:\\data\\pentx.mpd";
   MPDfile = fopen(mpdifilename, "rb");

   if (MPDfile == NULL)
     {
      cerr << "Could not open MPD file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The MPD file is opened" << endl;
     }
  }

void TDistanceTransform :: ReadImage(ImagePPtr& Pixel)
  {
   cout << "Reading the data file ..." << endl;

   for (r = 0; r < maxrow; r++)
     {
      fseek(MPDfile, r * sizeof(short) * MPI.Npixels, SEEK_SET);
      fread(Pixel[r], sizeof(short) * MPI.Npixels, 1, MPDfile);
     }
   cout << endl;
   fclose(MPDfile);
  }
```

```
void TDistanceTransform :: PrintPixel(ImagePPtr Pixel)
  {
   int r, c;
   for (r = 0; r < maxrow; r ++)
     {
      for (c = 0; c < maxcol; c ++)
        {
         cout << "Pixel value at [row][col]: [" << r << "][" << c << "]: "
              << Pixel[r][c] << "\r";
        }
     }
  }

void TDistanceTransform :: WriteMPIOutputFile()
  {
   //FILE* MPIofile;
   char mpiofilename[] = "c:\\data\\pentod.MPI";
   MPIofile = fopen(mpiofilename, "w");

   MpiOut.Nscanlines = MPI.Nscanlines;
   MpiOut.Npixels = MPI.Npixels;
   MpiOut.Pvmin = 0;
   MpiOut.Pvmax = 6000;
   MpiOut.Maptype = 2;
   MpiOut.Patch = MPI.Patch;
   MpiOut.Scale = 0;
   MpiOut.Crdtype = MPI.Crdtype;
   MpiOut.a11 = 0;
   MpiOut.a12 = 0;
   MpiOut.a21 = 0;
   MpiOut.a22 = 0;
   MpiOut.b1 = 0;
   MpiOut.b2 = 0;

   fwrite(&MpiOut, sizeof(MpiType), 1, MPIofile);
   cout << endl;
   cout << "Finished writing to file." << endl;
   fclose(MPIofile);
  }

void TDistanceTransform :: WriteMPDOutputFile()
  {
   //FILE* MPDofile;
   char mpdofilename[] = "c:\\data\\pentod.mpd";
   MPDofile = fopen(mpdofilename, "w+b");
   for (r = 0; r < maxrow; r ++)
     {
      fwrite(Pixel[r], sizeof(short) * MPI.Npixels, 1, MPDofile);
     }
   fclose(MPDofile);
  }

void TDistanceTransform :: SetBackground(ImagePPtr Pixel, int Bg, int Fg)
  {
   int row;
   int col;
   cout << "SetBackground ..." << endl;
   for (row = 0; row < maxrow; row ++)
     {
      for (col = 0; col < maxcol; col ++)
```

```
        {
         if (Pixel[row][col] < 0)
             Pixel[row][col] = Bg;
         else
         if (Fg > 0)
             Pixel[row][col] = Fg;

        }
     }
  }

void TDistanceTransform :: GetUpperMask(int r, int c, ImagePPtr Pixel, Mask&
MaskPix)
  {
   MaskPix[0] = Pixel[r-1][c-1];
   MaskPix[1] = Pixel[r-1][c];
   MaskPix[2] = Pixel[r-1][c+1];
   MaskPix[3] = Pixel[r][c-1];
   MaskPix[4] = Pixel[r][c];
  }

void TDistanceTransform :: GetLowerMask(int r, int c, ImagePPtr Pixel, Mask&
MaskPix)
  {
   MaskPix[4] = Pixel[r][c];
   MaskPix[5] = Pixel[r][c+1];
   MaskPix[6] = Pixel[r+1][c-1];
   MaskPix[7] = Pixel[r+1][c];
   MaskPix[8] = Pixel[r+1][c+1];
  }

int TDistanceTransform :: MinByIndex(int from, int to)
  {
   int i;
   int temp;
   int Idx;

   temp = MaskPix[from];
   Idx = from;
   for (i = from; i <= to ; i ++)
     {
      if (temp > MaskPix[i])
        {
         temp = MaskPix[i];
         Idx = i;
        }
     }
   return Idx;
  }

int TDistanceTransform :: Min5(int a, int b, int c, int d, int e)
  {
   int mn;
   mn = min(a, b);
   mn = min(mn, c);
   mn = min(mn, d);
   mn = min(mn, e);
   return mn;
  }
```

```
void TDistanceTransform :: DistancePassOne(ImagePPtr Pixel)
  {
   int r;
   int c;
   int k;
   for (r = 1; r < maxrow; r ++)
     {
      for (c = 1; c < maxcol; c ++)
        {
         GetUpperMask(r, c, Pixel, MaskPix);
         for (k = 0; k < 4; k ++)
           {
            if ((k == 0) || (k == 2))
              MaskPix[k] = MaskPix[k] + 4;
            else
              MaskPix[k] = MaskPix[k] + 3;
           }
         if (MaskPix[4] != 30000)
             MaskPix[4] = 0;

         Pixel[r][c] = MaskPix[MinByIndex(0, 4)];

        }
      }
  }

void TDistanceTransform :: DistancePassTwo(ImagePPtr Pixel)
  {
   int r;
   int c;
   int k;

   cout << endl;
   for (r = maxrow - 2; r > 0; r --)
     {
      for (c = maxcol - 2; c > 0; c --)
        {
         GetLowerMask(r, c, Pixel, MaskPix);
         for (k = 8; k > 4; k --)
           {
            if ((k == 6) || (k == 8))
              MaskPix[k] = MaskPix[k] + 4;
            else
              MaskPix[k] = MaskPix[k] + 3;
           }
         Pixel[r][c] = MaskPix[MinByIndex(4, 8)];
        }
     }
  }

void TDistanceTransform :: ForwardDistance()
  {
   cout << "Forward Distance ..." << endl;
   ReadImage(Pixel);
   SetBackground(Pixel, 30000, 0);
   DistancePassOne(Pixel);
  }

void TDistanceTransform :: BackwardDistance()
  {
```

```
   cout << "Backward Distance ..." << endl;
   DistancePassTwo(Pixel);
  }

void TDistanceTransform :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

void TDistanceTransform :: Title()
  {
   cout << endl;
  cout << "-----------Object-Oriented Distance Transformation --------------"
<< endl;
   cout << "----------- By: Alias Abdul-Rahman, November, 1998 -----" << endl;
   cout << " Input/Output : ILWIS .mpd, .mpi files. " << endl;
   cout << endl;
  }
```

2D Voronoi Tessellations

The following gives the complete codes for the 2D Voronoi Tessellations program.

```
//*********************************************************************//
//              A Voronoi Tesselation program for 2D TIN
//
//              Input  :  MPI and MPD files of DT program
//
//              Output :  MPI and MPD files
//
//              Copyright (c) Alias Abdul-Rahman, 1999
//
//*********************************************************************//

// File: TVoronoi.h

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <io.h>
#include <iomanip.h>
#include <dos.h>
#include <alloc.h>
#include <malloc.h>
#include <conio.h>
#include <limits.h>
#include <stddef.h>

#include "TDistanceTransform.h"

#define masksize 8                          // Mask size

class TVoronoiTessellation : public TDistanceTransform
  {
   public:
        TVoronoiTessellation();              // constructor
        ~TVoronoiTessellation();             // destructor

      Mask MaskPixV;
      FILE* MPDfile;
      FILE* MPDodfile;
      FILE* MPDovfile;
      FILE* MPIodfile;
      FILE* MPIovfile;
      ImagePPtr PixelV;

      // Function members
      void GetMPDInputFile();
      void ReadImage(ImagePPtr&);
      void CopyImage(ImagePPtr, ImagePPtr&);
```

```
    void GetMPIInputFile();
    void WriteMPIOutputFileDist();
    void WriteMPDOutputFileDist();
    void WriteMPIOutputFileVoronoi();
    void WriteMPDOutputFileVoronoi();
    void SetBackground(ImagePPtr, int, int);
    void GetUpperMaskDist(int, int, ImagePPtr, Mask&);
    void GetLowerMaskDist(int, int, ImagePPtr, Mask&);
    void GetUpperMaskVoronoi(int, int, ImagePPtr, Mask&);
    void GetLowerMaskVoronoi(int, int, ImagePPtr, Mask&);
    int MinByIndex(int, int);
    void ForwardPass(ImagePPtr, ImagePPtr);
    void BackwardPass(ImagePPtr, ImagePPtr);
    void ForwardVoronoi();
    void BackwardVoronoi();
    void PressAnyKey();
    void Title();
};
```

```
//*************************************************************************
//      Object-Oriented Voronoi Tesselation program for 2D TIN
//
//              Input  :  MPI and MPD files of DT program
//
//              Output :  MPI and MPD files
//
//      Copyright (c) Alias Abdul-Rahman, 1999
//
//*************************************************************************

// File: TVoronoi.cpp

#include "TVoronoi.h"

// main program
void main()
  {
   TVoronoiTessellation Voronoi;
   Voronoi.Title();
   TVoronoiTessellation();
   Voronoi.GetMPDInputFile();
   Voronoi.ForwardVoronoi();
   Voronoi.BackwardVoronoi();
   Voronoi.WriteMPDOutputFileDist();
   Voronoi.WriteMPIOutputFileDist();
   Voronoi.WriteMPDOutputFileVoronoi();
   Voronoi.WriteMPIOutputFileVoronoi();
  }

// Definition section follows:

// Constructor
TVoronoiTessellation :: TVoronoiTessellation()
 //: TDistanceTransform()
  {
   GetMPIInputFile();

   // Allocate memory dynamically for DT
   Pixel = new ImagePtr[maxrow];
   PixelV = new ImagePtr[maxrow];
   for (r = 0; r < maxrow; r ++)
     {
      Pixel[r] = new Image[maxcol];
     }

   // Allocate memory for Voronoi

   for (int j = 0; j < maxrow; j ++)
     {
      PixelV[j] = new Image[maxcol];
     }
  }

// Destructor
TVoronoiTessellation :: ~TVoronoiTessellation()
  {
   // Deallocate memory for DT
   for (r = 0; r < maxrow; r ++)
     {
```

```
      delete [] Pixel[r];
     }

   // Deallocate memory for Voronoi
   for (int j = 0; j < maxrow; j ++)
     {
      delete [] PixelV[j];
     }

   delete [] Pixel;
   delete [] PixelV;
  }

// Procedure: GetMPDFile
void TVoronoiTessellation :: GetMPDInputFile()
  {
   char mpdifilename [] = "c:\\data\\lbm.mpd";
   MPDfile = fopen(mpdifilename, "rb");

   if (MPDfile == NULL)
     {
      cerr << "Could not open MPD file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The MPD file is opened" << endl;
     }
  }

// Procedure: ReadImage
void TVoronoiTessellation :: ReadImage(ImagePPtr& Pixel)
  {
   cout << "Reading the data file ..." << endl;
   int r;
   for (r = 0; r < maxrow; r++)
     {
      fseek(MPDfile, r * sizeof(short) * MPI.Npixels, SEEK_SET);
      fread(Pixel[r], sizeof(short) * MPI.Npixels, 1, MPDfile);
     }
   cout << endl;
   fclose(MPDfile);
  }

// Procedure: CopyImage
void TVoronoiTessellation :: CopyImage(ImagePPtr PixA, ImagePPtr& PixB)
  {
   int j;
   int k;
   for (j = 0; j < maxrow; j ++)
     {
     for (k = 0; k < maxcol; k ++)
       {
        PixB[j][k] = PixA[j][k];
       }
     }

  }
```

```
void TVoronoiTessellation :: GetMPIInputFile()
  {
   FILE* MPIfile;
   char mpifilename[] = "c:\\data\\lbm.mpi";
   MPIfile = fopen(mpifilename, "rb");

   if (MPIfile == NULL)
     {
       cerr << " Could not open MPI file !" << endl;
       exit(1);
     }
   else
     {
      cout << "Information of  " << mpifilename << endl;
      fread(&MPI, sizeof(MpiType), 1, MPIfile);
      int size = sizeof(MpiType);

      maxrow = MPI.Nscanlines;
      maxcol = MPI.Npixels;

      cout << "MPI file size       : " << size << endl;
      cout << endl;
      cout << "Number of lines     : " << MPI.Nscanlines << endl;
      cout << "Number of columns   : " << MPI.Npixels << endl;
      cout << "Minimum value       : " << MPI.Pvmin << endl;
      cout << "Maximum value       : " << MPI.Pvmax << endl;
      cout << "Map type            : " << MPI.Maptype << endl;
      cout << "Patched             : " << MPI.Patch << endl;
      cout << "Scale (power of 10) : " << MPI.Scale << endl;
      cout << "Coordinate type     : " << MPI.Crdtype << endl;
      cout << "Transformation parameters :" << endl;
      cout << " a11  : " << MPI.a11 << endl;
      cout << " a12  : " << MPI.a12 << endl;
      cout << " a21  : " << MPI.a21 << endl;
      cout << " a22  : " << MPI.a22 << endl;
      cout << " b1   : " << MPI.b1 << endl;
      cout << " b2   : " << MPI.b2 << endl;
      fclose(MPIfile);
     }
  }


// Procedure: WriteMPIOutputFileDist
void TVoronoiTessellation :: WriteMPIOutputFileDist()
  {
   char mpiodfilename[] = "c:\\data\\lbmtryd.MPI";
   MPIodfile = fopen(mpiodfilename, "w");

   MpiOut.Nscanlines = MPI.Nscanlines;
   MpiOut.Npixels = MPI.Npixels;
   MpiOut.Pvmin = 0;
   MpiOut.Pvmax = 6000;
   MpiOut.Maptype = 2;
   MpiOut.Patch = MPI.Patch;
   MpiOut.Scale = 0;
   MpiOut.Crdtype = MPI.Crdtype;
   MpiOut.a11 = 0;
   MpiOut.a12 = 0;
   MpiOut.a21 = 0;
   MpiOut.a22 = 0;
   MpiOut.b1 = 0;
```

```
    MpiOut.b2 = 0;

    fwrite(&MpiOut, sizeof(MpiType), 1, MPIodfile);
    cout << endl;
    cout << "Finished writing to file." << endl;
    fclose(MPIodfile);
   }


// Procedure: WriteMPDOutputFileDist
void TVoronoiTessellation :: WriteMPDOutputFileDist()
   {
    int j;
    char mpdodfilename[] = "c:\\data\\lbmtryd.MPD";
    MPDodfile = fopen(mpdodfilename, "w+b");
    for (j = 0; j < maxrow; j ++)
      {
       fwrite(Pixel[j], sizeof(short) * MPI.Npixels, 1, MPDodfile);
      }
    fclose(MPDodfile);
   }


// Procedure: WriteMPIOutputVoronoiFile
void TVoronoiTessellation :: WriteMPIOutputFileVoronoi()
   {
    char mpiovfilename[] = "c:\\data\\lbmtryv.MPI";
    MPIovfile = fopen(mpiovfilename, "w");

    MpiOut.Nscanlines = MPI.Nscanlines;
    MpiOut.Npixels = MPI.Npixels;
    MpiOut.Pvmin = 1;
    MpiOut.Pvmax = 7000;
    MpiOut.Maptype = 2;
    MpiOut.Patch = MPI.Patch;
    MpiOut.Scale = 0;
    MpiOut.Crdtype = MPI.Crdtype;
    MpiOut.a11 = 0;
    MpiOut.a12 = 0;
    MpiOut.a21 = 0;
    MpiOut.a22 = 0;
    MpiOut.b1 = 0;
    MpiOut.b2 = 0;

    fwrite(&MpiOut, sizeof(MpiType), 1, MPIovfile);
    cout << endl;
    cout << "Finished writing to file." << endl;
    fclose(MPIovfile);
   }


// Procedure: WriteMPDOutputFileVoronoi
void TVoronoiTessellation :: WriteMPDOutputFileVoronoi()
   {
    int j;
    char mpdovfilename[] = "c:\\data\\lbmtryv.MPD";
    MPDovfile = fopen(mpdovfilename, "w+b");
    for (j = 0; j < maxrow; j ++)
      {
       fwrite(PixelV[j], sizeof(short) * MPI.Npixels, 1, MPDovfile);
      }
```

```
   fclose(MPDovfile);
  }

// Procedure: SetBackground
void TVoronoiTessellation :: SetBackground(ImagePPtr Pixel, int Bg, int Fg)
  {
   int row;
   int col;
   for (row = 0; row < maxrow; row ++)
     {
      for (col = 0; col < maxcol; col ++)
        {
         if (Pixel[row][col] <= 0)
             Pixel[row][col] = Bg;
         else
         if (Fg > 0)
             Pixel[row][col] = Fg;

        }
     }
  }

// Procedure: GetUpperMaskDist
void TVoronoiTessellation :: GetUpperMaskDist(int r, int c, ImagePPtr Pixel,
Mask& MaskPix)
  {
   MaskPix[0] = Pixel[r-1][c-1];
   MaskPix[1] = Pixel[r-1][c];
   MaskPix[2] = Pixel[r-1][c+1];
   MaskPix[3] = Pixel[r][c-1];
   MaskPix[4] = Pixel[r][c];
  }

// Procedure: GetLowerMaskDist
void TVoronoiTessellation :: GetLowerMaskDist(int r, int c, ImagePPtr Pixel,
Mask& MaskPix)
  {
   MaskPix[4] = Pixel[r][c];
   MaskPix[5] = Pixel[r][c+1];
   MaskPix[6] = Pixel[r+1][c-1];
   MaskPix[7] = Pixel[r+1][c];
   MaskPix[8] = Pixel[r+1][c+1];
  }

// Procedure: GetUpperMaskVoronoi
void TVoronoiTessellation :: GetUpperMaskVoronoi(int r, int c, ImagePPtr
PixelV, Mask& MaskPixV)
  {
   MaskPixV[0] = PixelV[r-1][c-1];
   MaskPixV[1] = PixelV[r-1][c];
   MaskPixV[2] = PixelV[r-1][c+1];
   MaskPixV[3] = PixelV[r][c-1];
   MaskPixV[4] = PixelV[r][c];
  }

// Procedure: GetLowerMaskVoronoi
void TVoronoiTessellation :: GetLowerMaskVoronoi(int r, int c, ImagePPtr
PixelV, Mask& MaskPixV)
  {
   MaskPixV[4] = PixelV[r][c];
   MaskPixV[5] = PixelV[r][c+1];
```

```
   MaskPixV[6] = PixelV[r+1][c-1];
   MaskPixV[7] = PixelV[r+1][c];
   MaskPixV[8] = PixelV[r+1][c+1];
  }

// Function: MinByIndex
int TVoronoiTessellation :: MinByIndex(int from, int to)
  {
   int i;
   int temp;
   int Idx;

   temp = MaskPix[from];
   Idx = from;
   for (i = from; i <= to ; i ++)
     {
      if (temp > MaskPix[i])
        {
         temp = MaskPix[i];
         Idx = i;
        }
     }
   return Idx;
  }

// Procedure: ForwardPass
void TVoronoiTessellation :: ForwardPass(ImagePPtr Pixel, ImagePPtr PixelV)
  {
   int r;
   int c;
   int k;
   for (r = 1; r < maxrow; r ++)
     {

      for (c = 1; c < maxcol; c ++)
        {
         GetUpperMaskDist(r, c, Pixel, MaskPix);
         GetUpperMaskVoronoi(r, c, PixelV, MaskPixV);
         for (k = 0; k < 4; k ++)
           {
            if ((k == 0) || (k == 2))
              MaskPix[k] = MaskPix[k] + 4;
            else
              MaskPix[k] = MaskPix[k] + 3;
           }
         if (MaskPix[4] != 255)
             MaskPix[4] = 0;

         Pixel[r][c] = MaskPix[MinByIndex(0, 4)];
         PixelV[r][c] = MaskPixV[MinByIndex(0, 4)];
        }
     }

  }

// Procedure: BackwardPass
void TVoronoiTessellation :: BackwardPass(ImagePPtr Pixel, ImagePPtr PixelV)
  {
   int r;
   int c;
```

```
    int k;

    cout << endl;
    for (r = maxrow - 2; r > 0; r --)
      {

       for (c = maxcol - 2; c > 0; c --)
         {
          GetLowerMaskDist(r, c, Pixel, MaskPix);
          GetLowerMaskVoronoi(r, c, PixelV, MaskPixV);
          for (k = 8; k > 4; k --)
            {
             if ((k == 6) || (k == 8))
               MaskPix[k] = MaskPix[k] + 4;
             else
               MaskPix[k] = MaskPix[k] + 3;
            }
          Pixel[r][c] = MaskPix[MinByIndex(4, 8)];
          PixelV[r][c] = MaskPixV[MinByIndex(4, 8)];
         }
      }
  }

// Procedure: ForwardVoronoi
void TVoronoiTessellation :: ForwardVoronoi()
  {
   cout << "Forward pass ..." << endl;
   ReadImage(Pixel);
   CopyImage(Pixel, PixelV);
   SetBackground(Pixel, 255, 0);
   ForwardPass(Pixel, PixelV);

  }

// Procedure: BackwardVoronoi
void TVoronoiTessellation :: BackwardVoronoi()
  {
   cout << "Backward pass ..." << endl;
   BackwardPass(Pixel, PixelV);

  }

// Procedure: PressAnyKey
void TVoronoiTessellation :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }


// Procedure: Title
void TVoronoiTessellation :: Title()
  {
   cout << endl;
    cout << "-----------Object-Oriented Distance Transformation-----------"
      << endl;
    cout << "-----------          and Voronoi Tessellation      -----------"
      << endl;
     cout << "----------- By: Alias Abdul-Rahman, November, 1998 -----" <<
```

```
    endl;
  cout << " Input/Output : ILWIS .mpd, .mpi files. " << endl;
  cout << endl;

  cout << endl;
  char anykey;
  cout << "Press any key to continue." << endl;
  getch();
}
```

2DTIN Data Structuring Code

The following gives the complete code for the 2D TIN data structure generation program.

```
//*************************************************************************//
//                 Object-Oriented Voronoi-to-TIN program
//
//                Input  :  MPI and MPD file of DT program
//
//                Output :  MPI and MPD files
//
//
//               Copyright (c) Alias Abdul-Rahman, 1999
//*************************************************************************//

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <io.h>
#include <iomanip.h>
#include <dos.h>
#include <alloc.h>
#include <malloc.h>
#include <conio.h>
#include <limits.h>
#include <stddef.h>

#include "TVoronoi.h"
//#include "TDistanceTransform.h"

// Constant section

#define masksize4 4                 // 2 x 2 Mask size

class TTinGeneration : public TVoronoiTessellation
  {
   public:
      TTinGeneration();             // constructor
      ~TTinGeneration();            // destructor

      typedef short DataType;
      typedef struct VertexStruct
                  {
                   DataType N1;
                   DataType N2;
                   DataType N3;
                  } TVertex;
```

```
    typedef struct TsNodeStruct
                {
                 short x;
                 short y;
                } TsNode;

    // data members
    typedef short Image;
    typedef Image* ImagePtr;
    typedef ImagePtr* ImagePPtr;
    typedef short Mask[4];

    //MpiType MPI;
    Mask MaskPix;
    ImagePPtr Pixel;
    FILE* MPDfile;
    FILE* MPIfile;

    FILE* TINfile;
    int i, r;
    int NTri;
    TVertex Element;
    TsNode TsPnt;
    bool FoundTri;

    // function members
    void GetMPDfile();
    void ReadImage(ImagePPtr&);
    void GetMPIfile();
    void GetTINfile();
    bool Less(DataType, DataType);
    bool Greater(DataType, DataType);
    void Swap(DataType&, DataType&);
    void NodeOrder(DataType&, DataType&, DataType&);
    void AddTritoFile(TVertex);
    void GetMask(int, int, ImagePPtr, Mask&);
    void GetSubImage(int, int, ImagePPtr, Mask&);
    void ScanlinesUp(Mask);
    void ScanlinesDown(Mask);
    void Scanlines(ImagePPtr);
    void MakeTIN();
    void Title();
    void PressAnyKey();
};
```

```
// Object-Oriented Program for Generating TIN
// File: TTinGen.cpp

#include "TTinGen.h"

// main program
void main()
  {
   TTinGeneration TIN;     // TIN is an object
   TIN.Title();
   TTinGeneration();       // constructor
   TIN.GetMPDfile();
   TIN.GetTINfile();
   TIN.MakeTIN();
   TIN.PressAnyKey();
  }


// Definitions section follows:
// A constructor
TTinGeneration :: TTinGeneration()
  {
   GetMPIfile();

   // Allocate memory dynamically
   Pixel = new ImagePtr[maxrow];
   for (r = 0; r < maxrow; r ++)
     {
      Pixel[r] = new Image[maxcol];
     }
  }

// A destructor
TTinGeneration :: ~TTinGeneration()
  {
   // Deallocate memory
   for (r = 0; r < maxrow; r ++)
     {
      delete [] Pixel[r];
     }

   delete [] Pixel;
  }


// Procedure: PresAnyKey
void TTinGeneration :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

// Procedure: GetMPDFile
void TTinGeneration :: GetMPDfile()
  {
   char mpdfilename [] = "c:\\data\\tryserv.mpd";
   MPDfile = fopen(mpdfilename, "rb");
```

```
    if (MPDfile == NULL)
      {
       cerr << "Could not open MPD file !" << endl;
       exit(1);
      }
    else
      {
       cout << endl;
       cout << "The MPD file is opened" << endl;
      }
  }

void TTinGeneration :: ReadImage(ImagePPtr& Pixel)
  {
   int r;
   for (r = 0; r < maxrow; r++)
     {
      fseek(MPDfile, r * sizeof(short) * MPI.Npixels, SEEK_SET);
      fread(Pixel[r], sizeof(short) * MPI.Npixels, 1, MPDfile);
     }
   cout << endl;
   fclose(MPDfile);
  }

void TTinGeneration :: GetMPIfile()
  {
   char mpifilename [] = "c:\\data\\tryserv.mpi";
   MPIfile = fopen(mpifilename, "rb");

   if (MPIfile == NULL)
     {
      cerr << "Could not open MPD file !" << endl;
      exit(1);
     }
   else
     {
      cout << "Information of  " << mpifilename << endl;
      fread(&MPI, sizeof(MpiType), 1, MPIfile);
      int size = sizeof(MpiType);

      maxrow = MPI.Nscanlines;
      maxcol = MPI.Npixels;

      cout << "MPI file size       : " << size << endl;
      cout << endl;
      cout << "Number of lines     : " << MPI.Nscanlines << endl;
      cout << "Number of columns   : " << MPI.Npixels << endl;
      cout << "Minimum value       : " << MPI.Pvmin << endl;
      cout << "Maximum value       : " << MPI.Pvmax << endl;
      cout << "Map type            : " << MPI.Maptype << endl;
      cout << "Patched             : " << MPI.Patch << endl;
      cout << "Scale (power of 10) : " << MPI.Scale << endl;
      cout << "Coordinate tyoe     : " << MPI.Crdtype << endl;
      cout << "Transformation parameters :" << endl;
      cout << " a11  : " << MPI.a11 << endl;
      cout << " a12  : " << MPI.a12 << endl;
      cout << " a21  : " << MPI.a21 << endl;
      cout << " a22  : " << MPI.a22 << endl;
      cout << " b1   : " << MPI.b1 << endl;
      cout << " b2   : " << MPI.b2 << endl;
      fclose(MPIfile);
```

```
      }
  }

// Procedure: GetTINfile
void TTinGeneration :: GetTINfile()
  {
   char TINfilename [] = "c:\\data\\tryser.tin";
   TINfile = fopen(TINfilename, "w");

   char* charN1 = "Node1";
   char* charN2 = "Node2";
   char* charN3 = "Node3";

   fprintf(TINfile, "%8s %8s %8s \n", charN1, charN2, charN3);
   if (TINfile == NULL)
     {
      cerr << "Could not open TIN file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The TIN file is okay for writing ... " << endl;
     }
   //fclose(TINfile);
  }

// Function: Less
bool TTinGeneration :: Less(DataType a, DataType b)
  {
   return (a < b);
  }

// Function: Greater
bool TTinGeneration :: Greater(DataType a, DataType b)
  {
   return (a > b);
  }

//Procedure: Swap
void TTinGeneration :: Swap(DataType& a, DataType& b)
  {
   DataType temp;

   temp = a;
   a = b;
   b = temp;
  }

// Procedure: NodeOrder
void TTinGeneration :: NodeOrder(DataType& a, DataType& b, DataType& c)
  {
   if (Greater(a, b))
       Swap(a, b);
   if (Greater(b, c))
       Swap(b, c);
   if (Greater(a, b))
       Swap(a, b);
  }
```

```
// Procedure: AddTritoFile
void TTinGeneration :: AddTritoFile(TVertex Triangle)
  {
   fprintf(TINfile, "%8d %8d %8d \n", Triangle.N1, Triangle.N2, Triangle.N3);
  }

//Procedure: GetSubImage
void TTinGeneration :: GetSubImage(int i, int c, ImagePPtr Pixel, Mask&
MaskPix)
  {
   int m, j;
   m = -1;
   for (i = 0; i < maxrow; i ++)
     {
      for (j = c; j < c + 1; j ++)
        {
         m ++;
         MaskPix[m] = Pixel[i][j];
        }
     }
  }

// Procedure: GetMask
void TTinGeneration :: GetMask(int r, int c, ImagePPtr Pixel, Mask& MaskPix)
  {
   MaskPix[0] = Pixel[r][c];
   MaskPix[1] = Pixel[r][c+1];
   MaskPix[2] = Pixel[r+1][c];
   MaskPix[3] = Pixel[r+1][c+1];
  }

// Procedure: ScanlinesUp
void TTinGeneration :: ScanlinesUp(Mask MaskPix)
  {
   bool DetectTri;
   bool NarrowBand;

   DetectTri = (MaskPix[0] != MaskPix[1]) &&
               (MaskPix[1] != MaskPix[2]) &&
               (MaskPix[2] != MaskPix[0]);

   NarrowBand = (MaskPix[0] == MaskPix[3]);
   NTri = 0;
   if (DetectTri && (! NarrowBand))
     {
      Element.N1 = MaskPix[0];
      Element.N2 = MaskPix[1];
      Element.N3 = MaskPix[2];
      NodeOrder(Element.N1, Element.N2, Element.N3);
      //NTri ++;
      AddTritoFile(Element);
      NTri ++;
      FoundTri = true;
     }
   else
     FoundTri = false;
  }

// Procedure: ScanlinesDown
void TTinGeneration :: ScanlinesDown(Mask MaskPix)
  {
```

```
   bool DetectTri;
   bool NarrowBand;

   DetectTri = (MaskPix[1] != MaskPix[2]) &&
               (MaskPix[2] != MaskPix[3]) &&
               (MaskPix[3] != MaskPix[1]);

   NarrowBand = (MaskPix[0] == MaskPix[3]);

   NTri = 0;
   if (DetectTri && (! NarrowBand))
      {
       Element.N1 = MaskPix[1];
       Element.N2 = MaskPix[2];
       Element.N3 = MaskPix[3];
       NodeOrder(Element.N1, Element.N2, Element.N3);
       //NTri ++;
       AddTritoFile(Element);
       NTri ++;
       FoundTri = true;
      }
   else
      FoundTri = false;
  }

// Procedure: Scanlines
void TTinGeneration :: Scanlines(ImagePPtr Pixel)
  {
   int r, c;
   for (r = 1; r < maxrow-1; r ++)
     {
      for (c = 1; c < maxcol-1; c ++)
        {
         GetMask(r, c, Pixel, MaskPix);
         ScanlinesUp(MaskPix);
         ScanlinesDown(MaskPix);
        }
     }
  }

// Procedure: MakeTIN
void TTinGeneration :: MakeTIN()
  {
   cout << "Reading the data ..." << endl;
   ReadImage(Pixel);
   cout << endl;
   cout << "Generating TIN ..." << endl;
   Scanlines(Pixel);
   cout << endl;
   cout << "Writing TIN output to file ... " << endl;
   fclose(TINfile);
  }

//Procedure: Title
void TTinGeneration :: Title()
  {
   cout << endl;
          cout    <<    "--------------    Object-Oriented    Voronoi-to-TIN
-----------------------" << endl;
   cout << "----------- By: Alias Abdul-Rahman, November, 1998 --------------"
<< endl;
```

```
  cout << "            Input : ILWIS .mpd, .mpi files. " << endl;
  cout << "            Output: TIN file (.tin), an ASCII file. " << endl;
  cout << endl;
}
```

```
// tinwin2d.rh

#define IDI_ICON1 1
#define CM_CLEAR           1125
#define CM_FILEEXIT        1126
#define CM_ABOUT           1127
#define CM_INPUTFILESITEM1    18874 // for XYZ file
#define CM_POPUPITEM1      18873 // for TIN file
```

2D Viewing Code

The following gives the complete code for the 2D viewing (SDI) in Object Windows Library (OWL) environment.

```
//*********************************************************//
// A Windows program for TIN display                       //
//    Input: XYZ coordinates and TIN file                  //
//    Output: TINs in Windows environment                  //
//    Copyright (c) Alias Abdul-Rahman , 1999              //
//*********************************************************//

#include <owl/pch.h>
#include <owl/applicat.h>
#include <owl/framewin.h>
#include <owl/decframe.h>
#include <owl/statusba.h>
#include <owl/controlb.h>
#include <owl/buttonga.h>
#include <owl/gdiobjec.h>
#include <owl/chooseco.h>
#include <owl/inputdia.h>
#include <owl/opensave.h>
#include <owl/dc.h>
#include <mem.h>

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <string.h>

#include "tinwin2d.rh"

#define maxtriangle 20000
#define maxpoint 8000
#define maxarc 200

class TTINDrawApp : public TApplication
  {
   public:
      TTINDrawApp() : TApplication() {}  // call base class constructor
      void InitMainWindow();
  };


class TTINWindow : public TWindow
  {
   public:
      TTINWindow(TWindow* parent = 0)
```

```
      : TWindow(parent) {}

    struct Point
      {
       float x;
       float y;
       float z;
      };

    struct Triangle
      {
       int Node1;
       int Node2;
       int Node3;
      };

    Point* pnt_ptr[maxpoint];
    Triangle* tri_ptr[maxtriangle];

    struct Polygon
      {
       int Snode;
       int Enode;
      };

    Polygon* poly_ptr[maxarc];

    FILE* XYZfile;
    FILE* TINfile;
    FILE* ARCfile;

    TOpenSaveDialog :: TData fileData;
    int k, i, t, ii;
    float xmin, xmax, ymin, ymax;
    float zmin, zmax;
    float xlength, ylength;
    int npnt, ntri, narc;
    int MaxX, MaxY;
    float scaleX, scaleY;

    void ReadXYZ();
    void ReadTIN();
    void ReadARC();
    void GetMinMax(float&, float&,
                   float&, float&,
                   float&, float&);
    void Paint(TDC& dc, bool, TRect&);
    void EvSize(uint, TSize&);
    void CmInputFilesItem1();
    void CmPopupItem1();
    void CmFileExit()
         {PostQuitMessage(0);}
    void CmClear();
    void CmAbout();
    DECLARE_RESPONSE_TABLE(TTINWindow);

  };

DEFINE_RESPONSE_TABLE1(TTINWindow, TWindow)
  EV_WM_SIZE,
```

```
  EV_COMMAND(CM_INPUTFILESITEM1, CmInputFilesItem1),
  EV_COMMAND(CM_POPUPITEM1, CmPopupItem1),
  EV_COMMAND(CM_ABOUT, CmAbout),
  EV_COMMAND(CM_FILEEXIT, CmFileExit),
  EV_COMMAND(CM_CLEAR, CmClear),
END_RESPONSE_TABLE;


void TTINWindow :: CmInputFilesItem1()
  {
   ReadXYZ();
  }

void TTINWindow :: CmPopupItem1()
  {
   ReadTIN();
  }

void TTINWindow :: ReadXYZ()
  {
   string charX, charY, charZ;

   // Allocate memory for XYZ points
   for (i = 0; i < maxpoint; i ++)
     {
      pnt_ptr[i] = new Point;
     }

   // Reads again the the XYZ file
   //ifstream XYZfile = "c:\\data\\lochgrd.xyz";
   //ifstream XYZfile = "c:\\data\\lochass.xyz";
   //ifstream XYZfile = "c:\\data\\ard.xyz";
   ifstream XYZfile = "c:\\data\\pent.xyz";

   if (! XYZfile)
     {
      MessageBox ("Unable to open file: XYZ file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);
     }
   else
     {
      // reads the file header
      XYZfile >> charX >> charY >> charZ;
      npnt = 0;
        do
         {
          npnt ++;
          XYZfile >> pnt_ptr[npnt] -> x
                  >> pnt_ptr[npnt] -> y
                  >> pnt_ptr[npnt] -> z;

         } while (! XYZfile.eof());

     //if (XYZfile.eof() == true)
        //MessageBox ("Finished read the XYZ points", "XYZ file complete",
                    //MB_OK | MB_ICONINFORMATION);
     }
   XYZfile.close();
  }

void TTINWindow :: ReadTIN()
```

```
  {
   string charNode1, charNode2, charNode3;

   // Allocate memory for Triangle
   for (t = 0; t < maxtriangle; t ++)
     {
      tri_ptr[t] = new Triangle;
     }

   // Reads the TIN file
   //ifstream TINfile = "c:\\data\\lochgrd.tin";
   //ifstream TINfile = "c:\\data\\lochas2.tin";
   //ifstream TINfile = "c:\\data\\ard.tin";
   ifstream TINfile = "c:\\data\\pent.tin";


   if (!TINfile)
     {
      MessageBox("Unable to open file: TIN file",
                 "File Error", MB_OK | MB_ICONEXCLAMATION);
     }

   else
     {
      ntri = 0;

      // reads the file header
      TINfile >> charNode1 >> charNode2 >> charNode3;
      do
        {
         ntri ++;
         TINfile >> tri_ptr[ntri] -> Node1
                 >> tri_ptr[ntri] -> Node2
                 >> tri_ptr[ntri] -> Node3;

        } while (! TINfile.eof());

       //if (TINfile.eof() == true)
        //MessageBox ("Finished read the TIN nodes", "TIN file complete",
                    //MB_OK | MB_ICONINFORMATION);
      }

   TINfile.close();
  }

void TTINWindow :: ReadARC()
  {
   string charSnode, charEnode;

   // Allocate memory for XYZ points
   for (int ii = 0; ii < maxarc; ii ++)
     {
      poly_ptr[ii] = new Polygon;
     }

   // Reads again the the XYZ file
   ifstream ARCfile = "c:\\data\\lake.arc";


   if (! ARCfile)
     {
```

```
      MessageBox ("Unable to open file: ARC file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);
    }
  else
    {
     // reads the file header
     ARCfile >> charSnode >> charEnode;
     narc = 0;
       do
        {
         narc ++;
         ARCfile >> poly_ptr[narc] -> Snode
                 >> poly_ptr[narc] -> Enode;

        } while (! ARCfile.eof());

    //if (ARCfile.eof() == true)
       //MessageBox ("Finished read the ARC points", "ARC file complete",
                  //MB_OK | MB_ICONINFORMATION);
    }
  ARCfile.close();
 }


void TTINWindow :: CmClear()
 {
  // to clear and redraw
  Invalidate();
 }


void TTINWindow :: CmAbout()
 {
  MessageBox ("TINs display with menu - by Alias Abdul-Rahman, 1999",
              "About the Program", MB_OK | MB_ICONINFORMATION);
 }

void TTINWindow :: GetMinMax(float& xmax, float& ymax,
                             float& xmin, float& ymin,
                             float& xlength, float& ylength)
 {
  string charX, charY, charZ;

  // Read the XYZ file and get the min-max of the coordinates
  //ifstream XYZfile = "c:\\data\\lochgrd.xyz";
  //ifstream XYZfile = "c:\\data\\lochass.xyz";
  //ifstream XYZfile = "c:\\data\\ard.xyz";
  ifstream XYZfile = "c:\\data\\pent.xyz";

  if (! XYZfile)
     MessageBox ("Unable to open file: XYZ file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);

  float x, y, z;

  // reads the file header
  XYZfile >> charX >> charY >> charZ;

  XYZfile >> x >> y >> z;
     xmin = x;
     xmax = x;
```

```
      ymin = y;
      ymax = y;

      zmin = z;
      zmax = z;

      do
        {
         XYZfile >> x >> y >> z;

         if (xmin > x)
             xmin = x;

         if (ymin > y)
             ymin = y;

         if (zmin > z)
             zmin = z;

         if (xmax < x)
             xmax = x;

         if (ymax < y)
             ymax = y;

         if (zmax < z)
             zmax = z;
         } while (! XYZfile.eof());

      xlength = xmax - xmin;
      ylength = ymax - ymin;
  }

void TTINWindow :: Paint(TDC& dc, bool, TRect&)
  {
   GetMinMax(xmax, ymax, xmin, ymin, xlength, ylength);

   ReadXYZ(); // reads the XYZ
   ReadTIN(); // reads the TIN
   //ReadARC(); // reads the ARC

   // to set the background color of the client area
   // comment out the next line for default colour
   //static const TColor color (RGB(0, 0, 0));      // RGB (0, 0, 0) - black
   //TWindow :: SetBkgndColor(color);               // RGB (192, 192, 192) -
gray


   RECT rect;
   :: GetClientRect(HWindow, &rect);

   dc.SetMapMode(MM_ANISOTROPIC);

   scaleX = (rect.right - rect.left) / xlength;
   scaleY = (rect.bottom - rect.top) / ylength;

   for (k = 1; k < npnt; k ++)
     {
      pnt_ptr[k] -> x = (pnt_ptr[k] -> x - xmin) * scaleX;
      pnt_ptr[k] -> y = (rect.bottom - rect.top) - ((pnt_ptr[k] -> y - ymin)
                                                    * scaleY);
```

```
  }

// set pen for points display
TPen pen1 = TPen(RGB(255, 0, 0), 1, PS_SOLID);
SelectObject(dc, pen1);
for (k = 1; k < npnt; k ++)
  {
   dc.MoveTo(pnt_ptr[k] -> x, pnt_ptr[k] -> y);
   dc.LineTo(pnt_ptr[k] -> x, pnt_ptr[k] -> y);
  }

SelectObject(dc, pen1);
DeleteObject(pen1);


// set pen style for TINs display
TPen pen2 = TPen(RGB(0, 0, 255), 1, PS_SOLID);
SelectObject(dc, pen2);

for (k = 1; k < ntri; k ++)
  {
   dc.MoveTo(pnt_ptr[tri_ptr[k] -> Node1] -> x,
             pnt_ptr[tri_ptr[k] -> Node1] -> y);

   dc.LineTo(pnt_ptr[tri_ptr[k] -> Node2] -> x,
             pnt_ptr[tri_ptr[k] -> Node2] -> y);
   dc.LineTo(pnt_ptr[tri_ptr[k] -> Node3] -> x,
             pnt_ptr[tri_ptr[k] -> Node3] -> y);
   dc.LineTo(pnt_ptr[tri_ptr[k] -> Node1] -> x,
             pnt_ptr[tri_ptr[k] -> Node1] -> y);
  }
SelectObject(dc, pen2);
DeleteObject(pen2);



// set pen style for polygon (i.e. constrained edges) display
TPen pen3 = TPen(RGB(255, 0, 0), 2, PS_SOLID);
SelectObject(dc, pen3);

for (int r = 1; r < narc; r ++)
  {
   dc.MoveTo(pnt_ptr[poly_ptr[r] -> Snode] -> x,
             pnt_ptr[poly_ptr[r] -> Snode] -> y);
   dc.LineTo(pnt_ptr[poly_ptr[r] -> Enode] -> x,
             pnt_ptr[poly_ptr[r] -> Enode] -> y);
   dc.LineTo(pnt_ptr[poly_ptr[r] -> Snode] -> x,
             pnt_ptr[poly_ptr[r] -> Snode] -> y);
  }
SelectObject(dc, pen3);
DeleteObject(pen3);



// Deallocate memory
for (i = 0; i < maxpoint; i ++)
  {
   delete [] pnt_ptr[i];
  }

for (t = 0; t < maxtriangle; t ++)
```

```
      {
       delete [] tri_ptr[t];
      }

    for (ii = 0; ii < maxarc; ii ++)
      {
       delete [] poly_ptr[ii];
      }

  }

// Force the window to repaint if resize
void TTINWindow :: EvSize(uint, TSize&)
  {
   :: InvalidateRect(HWindow, 0, true);
  }

void TTINDrawApp :: InitMainWindow()
  {
   // construct the decorated MDI frame window
   TDecoratedFrame* frame = new TDecoratedFrame(0,
                      "TINSoft ver. 1.0", new TTINWindow);

   // construct the status bar
   TStatusBar* sb = new TStatusBar(frame, TGadget :: Recessed,
                                   TStatusBar :: CapsLock | TStatusBar ::
NumLock);

   // construct the control bar
   TControlBar* cb = new TControlBar(frame);

   cb -> Insert(*new TButtonGadget(CM_INPUTFILESITEM1, CM_INPUTFILESITEM1,
                      TButtonGadget :: Command));
   cb -> Insert(*new TButtonGadget(CM_POPUPITEM1, CM_POPUPITEM1,
                      TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_CLEAR, CM_CLEAR,
                      TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_FILEEXIT, CM_FILEEXIT,
                      TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_ABOUT, CM_ABOUT,
                      TButtonGadget :: Command));

   cb -> SetHintMode(TGadgetWindow :: EnterHints);

   // set client area to the application workspace
   frame -> SetBkgndColor(::GetSysColor(COLOR_APPWORKSPACE));

   // insert the status and control bar into the frame
   frame -> Insert(*sb, TDecoratedFrame :: Bottom);
   frame -> Insert(*cb, TDecoratedFrame :: Top);

   // set the main window and its menu
```

```
    SetMainWindow(frame);
    GetMainWindow() -> AssignMenu("COMMANDS");
    EnableCtl3d(true);
   }


int OwlMain(int /*argc*/, char* /*argv*/ [])
   {
    return TTINDrawApp().Run();
   }
```

```
#include "tinwin2d.rh"

COMMANDS MENU
{
 POPUP "&InputFiles"
 {
  MENUITEM "&XYZ File", CM_INPUTFILESITEM1
  MENUITEM "&TIN File", CM_POPUPITEM1
 }

 POPUP "&Redraw"
 {
  MENUITEM "&Clear and Redraw", CM_CLEAR
  MENUITEM "E&xit", CM_FILEEXIT
 }

 MENUITEM "&About", CM_ABOUT
}

STRINGTABLE
  {
   CM_INPUTFILESITEM1,   "Get the XYZ file"
   CM_POPUPITEM1,        "Get the TIN file"
   CM_CLEAR,             "Redraw the TINs"
   CM_FILEEXIT,          "Exit the program ..."
   CM_ABOUT,             "About the program ..."
  }

CM_CLEAR BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 91 77 77 77 77 77 77 77 77 99'
 '00 00 79 77 77 77 77 77 77 77 79 17 00 00 77 17'
 '77 87 77 77 77 79 91 77 00 00 77 77 80 08 88 87'
 '77 77 77 77 00 00 77 78 00 00 08 88 77 77 77 77'
 '00 00 77 78 0A 0A A0 08 88 87 77 77 00 00 77 78'
 '0A 0A AA A0 88 88 77 77 00 00 77 80 AA 0A AA AA'
 '00 88 77 77 00 00 77 80 AA A0 AA A0 A0 88 77 77'
 '00 00 77 0A AA A0 AA 0A A0 88 71 17 00 00 77 0A'
 'AA A0 00 AA A0 88 79 97 00 00 78 0A AA A0 AA AA'
 'A0 88 77 77 00 00 78 00 00 00 AA AA A0 88 77 77'
 '00 00 78 88 87 78 0A AA A0 88 77 77 00 00 77 77'
 '77 77 80 AA A0 88 77 77 00 00 77 71 77 77 78 0A'
 'A0 88 77 77 00 00 77 19 77 77 77 80 A0 88 77 77'
 '00 00 77 97 77 77 77 78 00 88 79 77 00 00 71 77'
 '77 77 77 77 88 87 71 19 00 00 19 77 77 77 77 77'
 '77 77 77 71 00 00'
}


CM_FILEEXIT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
```

```
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 EE EE EE EE EE EE EE EE EE EE'
 '00 00 EE EE FF EF EF FE EE EE EE EE 00 00 EE EE'
 'FF EE FE EF EE EE EE EE 00 00 EE EE FF EF FF FF'
 'EE EE EE EE 00 00 EE EE EF EE EF FF EE EE EE EE'
 '00 00 88 88 88 88 88 88 88 88 88 88 00 00 66 66'
 '8A AA A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A A9 A8 77 86 66 66 66 00 00 66 66'
 '8A A9 A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A AA A8 77 86 66 66 66 00 00 66 66'
 '8A A8 87 77 86 66 66 66 00 00 66 66 8A 87 77 77'
 '86 66 66 66 00 00 66 66 88 77 77 77 86 66 66 66'
 '00 00 66 66 88 88 88 88 86 66 66 66 00 00 66 66'
 '66 66 66 66 66 66 66 66 00 00 66 66 66 66 66 66'
 '66 66 66 66 00 00'
}


CM_ABOUT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB B8 88 88 BB BB BB BB'
 '00 00 BB BB BB B8 00 88 BB BB BB BB 00 00 BB BB'
 'BB B8 00 BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00 BB BB BB B8 00 88 BB BB BB BB'
 '00 00 BB BB BB BB 80 08 BB BB BB BB 00 00 BB BB'
 'B8 8B B8 00 8B BB BB BB 00 00 BB BB B8 00 BB 00'
 '8B BB BB BB 00 00 BB BB B8 00 BB 00 8B BB BB BB'
 '00 00 BB BB BB 80 00 00 8B BB BB BB 00 00 BB BB'
 'BB B8 00 00 8B BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00'
}

CM_INPUTFILESITEM1 BITMAP "pointfile.bmp"
/*{
 '42 4D 96 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 18 00 00 00 18 00 00 00 01 00 04 00 00 00'
 '00 00 20 01 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
```

```
 '00 00 FF FF FF 00 FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF 00 00 00 00 00 00 00 00 00 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 0A AA AA AA AA'
 'AA AA AA AA 08 8F FF 0A 99 9A AA AA AA AA AA AA'
 '08 8F FF 0A AA AA AA AA AA AA AA AA 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 00 00 00 00 00'
 '00 00 00 00 08 8F FF 88 88 88 88 88 88 88 88 88'
 '88 8F FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF F0 00 00 00 00'
 '00 00 0F FF FF FF FF F0 FF FF FF FF FF FF 0F FF'
 'FF FF FF F0 F0 00 FF FF FF FF 0F FF FF FF FF F0'
 'FF FF FF F0 00 0F 0F FF FF FF FF F0 FF FF FF FF'
 'FF FF 0F FF FF FF FF F0 FF FF FF 00 0F FF 0F FF'
 'FF FF FF F0 FF FF FF FF FF F0 8F FF FF FF FF F0'
 'F0 00 FF FF FF 08 8F FF FF FF FF F0 FF FF FF FF'
 '00 88 8F FF FF FF FF F0 FF FF FF F0 88 88 8F FF'
 'FF FF FF F0 00 00 00 08 88 88 8F FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF'
}
*/

CM_POPUPITEM1 BITMAP "tinfile.bmp"
/*{
 '42 4D 96 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 18 00 00 00 18 00 00 00 01 00 04 00 00 00'
 '00 00 20 01 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF 00 00 00 00 00 00 00 00 00 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 0A AA AA AA AA'
 'AA AA AA AA 08 8F FF 0A A9 99 9A AA AA AA AA AA'
 '08 8F FF 0A AA AA AA AA AA AA AA AA 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 00 00 00 00 00'
 '00 00 00 00 08 8F FF 88 88 88 88 88 88 88 88 88'
 '88 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00'
 '00 00 00 FF FF FF FF FF 0F FF FF FF FF FF F0 FF'
 'FF FF FF FF 0F FF FF FF 00 FF F0 FF FF FF FF FF'
 '0F FF FF FF FF FF F0 FF FF FF FF FF 0F F0 00 0F'
 'FF FF F0 FF FF FF FF FF 0F FF FF FF FF FF F0 FF'
 'FF FF FF FF 0F FF FF F0 0F FF F0 FF FF FF FF FF'
 '0F FF FF FF FF FF 08 FF FF FF FF FF 0F F0 00 FF'
 'FF F0 88 FF FF FF FF FF 0F FF FF FF F0 08 88 FF'
 'FF FF FF FF 0F FF FF FF 08 88 88 FF FF FF FF FF'
 '00 00 00 00 88 88 88 FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF'
}

*/
```

3D Rasterization Code

The following gives the complete code for the 3D rasterization program.

```
//*************************************************************************//
// A program to rasterise points in 3D                                     //
//      Input: XYZ coordinate                                              //
//      Output: 3D array ILWIS look-like .MPD and .MPI file format         //
//                                                                         //
// ************************************************************************//

#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <iomanip.h>
#include <io.h>
#include <conio.h>
#include <string>


const int rowsize = 190;
const int colsize = 190;
const int zsize   = 190;

typedef struct PointStruct            // structure for XYZ world coordinates
  {
   float x;
   float y;
   float z;
  } PointType;

PointType Point;

typedef struct VpiStruct
  {
   short Nscanlines;       // no. of image rows
   short Npixels;          // no. of image cols
   short Nlevels;          // no. of image levels
   short Pvmin;            // voxel's minimum value
   short Pvmax;            // voxel's maximum value
   short Maptype;          // map type
   short Patch;            // patch type
   short Scale;            // image scale factor
   short Crdtype;          // coordinate type
   float a11;              // transformation cooefficient (3 x 3 matrix)
   float a12;              //                  "
   float a13;              //                  "
   float a21;              //                  "
   float a22;              //                  "
   float a23;              //                  "
   float a31;              //                  "
```

```
   float a32;              //                  "
   float a33;              //                  "
   float b1;               // translation vector (3 x 1 matrix)
   float b2;               //                  "
   float b3;               //                  "
  } VpiType;

VpiType VPI;

// typedef section
typedef short Image[rowsize][colsize][zsize];    // Image: Voxel variable

Image Voxel;

float xlength, ylength;
float zlength;
int voxelsize;
short maxrow, maxcol, maxlevel;
short npnt, pnt, maxpnt;
float scalex, scaley, scalez, scale;
float xmin, ymin, xmax, ymax;
float zmin, zmax;
FILE* VPDfile;
FILE* VPIfile;


// prototype section
void GetXYZfile(float&, float&, float&, float&, float&, float&);
void WriteVpiFile();
void GetVoxelSize(int&);
void GetRowColLevel(int, float&, float&, float&);
void Make3DRaster(PointType, int, float, float, float);
void DisplayVoxel();
void PressAnyKey();

// Main
void main()
  {
   GetXYZfile(xmin, ymin, zmin, xmax, ymax, zmax);
   GetVoxelSize(voxelsize);
   GetRowColLevel(voxelsize, xlength, ylength, zlength);
   Make3DRaster(Point, voxelsize, xlength, ylength, zlength);
   WriteVpiFile();
   fclose(VPDfile);
   fclose(VPIfile);
   PressAnyKey();
  }

// definition section
void GetXYZfile(float& xmin, float& ymin, float& zmin,
                float& xmax, float& ymax, float& zmax)
  {
   ifstream XYZfile("c:\\data\\lakebore.xyz");

     if (! XYZfile)
       {
        cerr << "Point file not found" << endl;
        exit(1);
       }
```

```
    npnt = 0;

    // reads the file header
    string charX, charY, charZ;
    XYZfile >> charX >> charY >> charZ;
    XYZfile >> Point.x >> Point.y >> Point.z;

    // to calculate min xyz, max xyz
    xmin = Point.x;
    ymin = Point.y;
    zmin = Point.z;

    xmax = Point.x;
    ymax = Point.y;
    zmax = Point.z;

    do
      {
       npnt ++;
       printf("\r");
       cout << "Reading points ... " << npnt;

       XYZfile >> Point.x >> Point.y >> Point.z;
         if (xmin > Point.x)
             xmin = Point.x;

         if (ymin > Point.y)
             ymin = Point.y;

         if (zmin > Point.z)
             zmin = Point.z;

         if (xmax < Point.x)
             xmax = Point.x;

         if (ymax < Point.y)
             ymax = Point.y;

         if (zmax < Point.z)
             zmax = Point.z;

      } while (! XYZfile.eof());
    npnt = npnt - 1;

    xlength = xmax - xmin;
    ylength = ymax - ymin;
    zlength = zmax - zmin;

    XYZfile.close();
   }

void GetVoxelSize(int& voxelsize)
   {
    cout << endl;
    cout << "Enter voxel size (in meter): " << endl;
    cin >> voxelsize;
   }

void GetRowColLevel(int voxelsize,
                    float& xlength, float& ylength, float& zlength)
   {
```

```
   xlength = xmax - xmin;
   ylength = ymax - ymin;
   zlength = zmax - zmin;

   maxrow   = int (ylength / voxelsize) + 1;
   maxcol   = int (xlength / voxelsize) + 1;
   maxlevel = int (zlength / voxelsize) + 1;
  }

void Make3DRaster(PointType Point, int voxelsize,
                  float xlength, float ylength, float zlength)
  {
   short PointID;
   int row, col, level;

   scale = 1.0 / float(voxelsize);

   cout << endl;
   cout << "xmin, ymin, zmin: " << xmin << ", " << ymin << ", " << zmin <<
endl;
   cout << "xmax, ymax, zmax: " << xmax << ", " << ymax << ", " << zmax <<
endl;
   cout << "voxel size : " << voxelsize << endl;
   cout << "xlength : " << xlength << endl;
   cout << "ylength : " << ylength << endl;
   cout << "zlength : " << zlength << endl;
   cout << "maxrow  :" << maxrow << endl;
   cout << "maxcol  :" << maxcol << endl;
   cout << "maxzrow :" << maxlevel << endl;

   cout << setiosflags(ios::showpoint) << endl;
   cout << setprecision(5) << "scale   : " << scale << endl;

   ifstream XYZfile("c:\\data\\lakebore.xyz");

   if (! XYZfile)
       {
        cerr << "Point file not found" << endl;
        exit(1);
       }

   string charX, charY, charZ;
   FILE* VPDfile;
   char vpdfilename[] = "c:\\data\\lakebo.vpd";
   VPDfile = fopen(vpdfilename, "wb");

   pnt = -1;

   // reads the file header
   XYZfile >> charX >> charY >> charZ;
   while (! XYZfile.eof())
     {
      pnt ++;
      printf("\r");
      cout << "Rasterizing the points ... " << pnt;
      XYZfile >> Point.x >> Point.y >> Point.z;
      col  = int (scale * (Point.x - xmin)) + 1;
      row  = int (scale * (Point.y - ymin)) + 1;
      //level = int ((-1 * scale) * (Point.z - zmin)) + maxlevel + 1;
      level = int (scale * (Point.z - zmin)) + 1;
      PointID = short(pnt-1);
```

```
      Voxel[level][row][col] = PointID;
     }

/*   do
     {
      printf("\r");
      cout << "Rasterizing the points ... " << pnt;
      XYZfile >> Point.x >> Point.y >> Point.z;
      PointID = short(pnt);
      col   = int (scale * (Point.x - xmin)) + 1;
      row   = int (scale * (Point.y - ymin)) + 1;
      level = int ((-1 * scale) * (Point.z - zmin)) + maxlevel + 1;
      Voxel[level][row][col] = PointID;
      pnt ++;
     } while (! XYZfile.eof());
*/
   for (level = 0; level < maxlevel; level ++)
     {
      for (row = 0; row < maxrow; row ++)
        {
         for (col = 0; col < maxcol; col ++)
           {
            fwrite(&Voxel[level][row][col], sizeof(short), 1, VPDfile);
           }
        }
     }

   cout << endl;
   cout << endl;
   cout << "Total points : " << npnt;
   cout << endl;
   cout << endl;
   XYZfile.close();
  }

void DisplayVoxel()
  {
   int l, r, c;

   cout << "Displaying voxel value ..." << endl;
   cout << endl;

   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         for (c = 0; c < maxcol; c ++)
           {
            printf("\r");
            cout << "Voxel[zrow][row][col] : Voxel[" << l << "][" << r <<
                 "][" << c << "] : " << Voxel[l][r][c];
           }
        }
     }


  }

void WriteVpiFile()
  {
   FILE* VPIfile;
```

```
    char vpifilename[] = "c:\\data\\lakebo.vpi";
    VPIfile = fopen(vpifilename, "wb");

    // assign the value for the VPI file
    VPI.Nscanlines = maxrow;
    VPI.Npixels = maxcol;
    VPI.Nlevels = maxlevel;
    VPI.Pvmin = 0;
    VPI.Pvmax = npnt;
    VPI.Maptype = 2;
    VPI.Patch = 0;
    VPI.Scale = 0;
    VPI.Crdtype = 1;
    VPI.a11 = scale;
    VPI.a12 = 0;
    VPI.a13 = 0;
    VPI.a21 = 0;
    VPI.a22 = scale;
    VPI.a23 = 0;
    VPI.a31 = 0;
    VPI.a32 = 0;
    VPI.a33 = -scale;
    VPI.b1 = 1;
    VPI.b2 = 1;
    VPI.b3 = maxlevel + 1;

    // write out the above values to the VPI file
    fwrite(&VPI, sizeof(VpiType), 1, VPIfile);
   }

void PressAnyKey()
   {
    cout << endl;
    cout << endl;
    cout << "Press any key to continue" << endl;
    getch();
    cout << endl;
   }
```

3D Distance Transformation Code

The following gives the complete codes for the 3D Distance Transformation program.

```
//*************************************************************************
**//
//      Object-Oriented 3D Distance Transformation (DT) program for 3D TIN
//
//              Input   :  3D array format
//
//              Output  :  3D array ILWIS look-like file format
//
//
//
//*************************************************************************
**//

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <io.h>
#include <iomanip.h>
#include <dos.h>
#include <malloc.h>
#include <conio.h>
#include <limits.h>
#include <stddef.h>

// Constant section

#define masksize 27             // Mask size

class DistanceTransform3D
  {
   public:
        DistanceTransform3D();
        ~DistanceTransform3D();

       // Data members
       // VPI input file structure
       typedef struct VpiStruct
                 {
                  short Nscanlines;       // no. of image rows
                  short Npixels;          // no. of image cols
                  short Nlevels;          // no. of image levels
                  short Pvmin;            // voxel's minimum value
                  short Pvmax;            // voxel's maximum value
                  short Maptype;          // map type
                  short Patch;            // patch type
                  short Scale;            // image scale factor
```

```
                short Crdtype;          // coordinate type
                float a11;              // transformation cooefficient (3 x
3 matrix)
                float a12;              //                  "
                float a13;              //                  "
                float a21;              //                  "
                float a22;              //                  "
                float a23;              //                  "
                float a31;              //                  "
                float a32;              //                  "
                float a33;              //                  "
               float b1;               // translation vector (3 x 1 matrix)
                float b2;               //                  "
                float b3;               //                  "
               } VpiType;

      VpiType VPI, VPIOut;

      typedef short Image;
      typedef Image* VoxelRow;
      typedef VoxelRow* Voxel2D;
      typedef Voxel2D* Voxel3D;

      typedef short Mask[masksize];

      // Global variables
      Mask MaskPix;
      FILE* VPDifile;
      FILE* VPDofile;
      FILE* VPIifile;
      FILE* VPIofile;

      Voxel3D Voxel;           // variable for voxels

      int maxrow;
      int maxcol;
      int maxlevel;
      float scale;
      int row, col, level;
      int r, l;

      // Function members
      void GetVPDInputFile();
      void GetVPIInputFile(VpiType& VPI);
      void ReadVoxelImage(Voxel3D&);
      void PrintVoxel(Voxel3D);
      void WriteVPDOutputFile();
      void WriteVPIOutputFile();
      void SetBackground(Voxel3D, int, int);
      void GetUpperMask(int, int, int, Voxel3D, Mask&);
      void GetLowerMask(int, int, int, Voxel3D, Mask&);
      int MinByIndex(int, int);
      int Min5(int, int, int, int, int);
      void DistancePassOne(Voxel3D);
      void DistancePassTwo(Voxel3D);
      void ForwardDistance();
      void BackwardDistance();
      void DisplayVoxel(Voxel3D);
      void PressAnyKey();
```

```
        void Title();
    };

// Main program
void main()
  {
   DistanceTransform3D ThreeDDT;     // ThreeDDT is an object
   ThreeDDT.Title();
   ThreeDDT.GetVPDInputFile();
   ThreeDDT.ForwardDistance();
   ThreeDDT.BackwardDistance();
   ThreeDDT.WriteVPDOutputFile();
   ThreeDDT.WriteVPIOutputFile();
   ThreeDDT.PressAnyKey();
  }

// End of main program

// Definitions sections follows:
// The constructor
DistanceTransform3D :: DistanceTransform3D()
  {
   GetVPIInputFile(VPI);

   // Allocate memory dynamically
   Voxel = new Voxel2D[maxlevel];
   for (l = 0; l < maxlevel; l ++)
     {
      Voxel[l] = new VoxelRow[maxrow];
      for (r = 0; r < maxrow; r ++)
        {
         Voxel[l][r] = new Image[maxcol];
        }
     }
  }

// The destructor
DistanceTransform3D :: ~DistanceTransform3D()
  {
   // Deallocate memory
   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         delete [] Voxel[l][r];
        }
     }
   delete [] Voxel;
  }

// Procedure: PressAnyKey
void DistanceTransform3D :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }


// Procedure: GetVoxelInputFile
```

```
void DistanceTransform3D :: GetVPDInputFile()
  {
   char vpdifilename [] = "c:\\data\\ras3d.vpd";

   VPDifile = fopen(vpdifilename, "rb");
   if (VPDifile == NULL)
     {
      cerr << "Could not open VPD file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The VPD file is opened" << endl;
     }
  }

// Procedure: GetVPIInputFile
void DistanceTransform3D :: GetVPIInputFile(VpiType& VPI)
  {
   char vpiifilename[] = "c:\\data\\ras3d.vpi";
   VPIifile = fopen(vpiifilename, "rb");
   if (VPIifile == NULL)
     {
      cerr << "Could not open VPI file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "Information of  " << vpiifilename << endl;
      fread(&VPI, sizeof(VpiType), 1, VPIifile);
      int size = sizeof(VpiType);

      maxrow   = VPI.Nscanlines;
      maxcol   = VPI.Npixels;
      maxlevel = VPI.Nlevels;

      cout << "VPI file size       : " << size << endl;
      cout << endl;
      cout << "Number of lines     : " << VPI.Nscanlines << endl;
      cout << "Number of columns   : " << VPI.Npixels << endl;
      cout << "Number of levels    : " << VPI.Nlevels << endl;
      cout << "Minimum value       : " << VPI.Pvmin << endl;
      cout << "Maximum value       : " << VPI.Pvmax << endl;
      cout << "Map type            : " << VPI.Maptype << endl;
      cout << "Patched             : " << VPI.Patch << endl;
      cout << "Scale (power of 10) : " << VPI.Scale << endl;
      cout << "Coordinate type     : " << VPI.Crdtype << endl;
      cout << "Transformation parameters :" << endl;
      cout << "    a11  : " << VPI.a11 << endl;
      cout << "    a12  : " << VPI.a12 << endl;
      cout << "    a13  : " << VPI.a31 << endl;
      cout << "    a21  : " << VPI.a21 << endl;
      cout << "    a22  : " << VPI.a22 << endl;
      cout << "    a23  : " << VPI.a23 << endl;
      cout << "    a31  : " << VPI.a31 << endl;
      cout << "    a32  : " << VPI.a32 << endl;
      cout << "    a33  : " << VPI.a33 << endl;
      cout << endl;
      cout << "Translation vectors : " << endl;
```

```
      cout << "   b1    : " << VPI.b1 << endl;
      cout << "   b2    : " << VPI.b2 << endl;
      cout << "   b3    : " << VPI.b3 << endl;
      fclose(VPIifile);
     }
  }

// Procedure: ReadVoxelImage
void DistanceTransform3D :: ReadVoxelImage(Voxel3D& Voxel)
  {
   int r;
   int c;
   int l;

   cout << endl;
   cout << "Reading voxels data ..., wait !" << endl;
   cout << endl;

   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         for (c = 0; c < maxcol; c ++)
           {
            fread(&Voxel[l][r][c], sizeof(short), 1, VPDifile);
           }
        }
     }
   cout << endl;
   fclose(VPDifile);
  }

// Procedure: PrintPixel
void DistanceTransform3D :: PrintVoxel(Voxel3D Voxel)
  {
   int r;
   int c;
   int l;

   for (r = 0; r < maxrow; r ++)
     {
      for (c = 0; c < maxcol; c ++)
        {
         for (l = 0; l < maxlevel; l ++)
           {
            cout << "Voxel value at [row][col][level]: [" << r << "][" << c
                 << "][" << l << "]: " << Voxel[r][c][l] << "\r";
           }
        }
     }
  }


// Procedure: WriteVoxelOutputFile
void DistanceTransform3D :: WriteVPDOutputFile()
  {
   int r;
   int l;
   int c;
```

```
    char vpdofilename[] = "c:\\data\\voxodcc.vpd";
    VPDofile = fopen(vpdofilename, "w+b");
    for (l = 0; l < maxlevel; l ++)
      {
       for (r = 0; r < maxrow; r ++)
         {
          for (c = 0; c < maxcol; c ++)
            {
             fwrite(&Voxel[l][r][c], sizeof(short), 1, VPDofile);
            }
         }
      }
    fclose(VPDofile);
  }

// Procedure: WriteVPIFile
void DistanceTransform3D :: WriteVPIOutputFile()
  {
   FILE* VPIofile;
   char vpifilename[] = "c:\\data\\voxodcc.vpi";
   VPIofile = fopen(vpifilename, "wb");

   // assign the value for the VPI file
   VPIOut.Nscanlines = VPI.Nscanlines;
   VPIOut.Npixels = VPI.Npixels;
   VPIOut.Nlevels = VPI.Nlevels;
   VPIOut.Pvmin = 0;
   VPIOut.Pvmax = 6000;
   VPIOut.Maptype = 2;
   VPIOut.Patch = 0;
   VPIOut.Scale = 0;
   VPIOut.Crdtype = 1;
   VPIOut.a11 = VPI.a11;
   VPIOut.a12 = 0;
   VPIOut.a13 = 0;
   VPIOut.a21 = 0;
   VPIOut.a22 = VPI.a22;
   VPIOut.a23 = 0;
   VPIOut.a31 = 0;
   VPIOut.a32 = 0;
   VPIOut.a33 = VPI.a33;
   VPIOut.b1 = 1;
   VPIOut.b2 = 1;
   VPIOut.b3 = maxlevel + 1;

   // write out the above values to the VPI file
   fwrite(&VPIOut, sizeof(VpiType), 1, VPIofile);
   fclose(VPIofile);
  }


// Procedure: SetBackground
void DistanceTransform3D :: SetBackground(Voxel3D Voxel, int Bg, int Fg)
  {
   int row;
   int col;
   int l;

   for (l = 0; l < maxlevel; l ++)
     {
      for (row = 0; row < maxrow; row ++)
```

```
        {
         for (col = 0; col < maxcol; col ++)
           {
            if (Voxel[l][row][col] == 0)
                Voxel[l][row][col] = Bg;
            else
            if (Fg > 0)
                Voxel[l][row][col] = Fg;
            }

        }
     }
  }

// Procedure: GetUpperMask
void DistanceTransform3D :: GetUpperMask(int l, int r, int c, Voxel3D Voxel,
Mask& MaskPix)
  {
   MaskPix[0]  = Voxel[l-1][r-1][c-1];
   MaskPix[1]  = Voxel[l-1][r-1][c];
   MaskPix[2]  = Voxel[l-1][r-1][c+1];
   MaskPix[3]  = Voxel[l-1][r][c-1];
   MaskPix[4]  = Voxel[l-1][r][c];
   MaskPix[5]  = Voxel[l-1][r][c+1];
   MaskPix[6]  = Voxel[l-1][r+1][c-1];
   MaskPix[7]  = Voxel[l-1][r+1][c];
   MaskPix[8]  = Voxel[l-1][r+1][c+1];
   MaskPix[9]  = Voxel[l][r-1][c-1];
   MaskPix[10] = Voxel[l][r-1][c];
   MaskPix[11] = Voxel[l][r-1][c+1];
   MaskPix[12] = Voxel[l][r][c-1];
   MaskPix[13] = Voxel[l][r][c];

  }

// Procedure: GetLowerMask
void DistanceTransform3D :: GetLowerMask(int l, int r, int c, Voxel3D Voxel,
Mask& MaskPix)
  {
   MaskPix[13] = Voxel[l][r][c];
   MaskPix[14] = Voxel[l][r][c+1];
   MaskPix[15] = Voxel[l][r+1][c-1];
   MaskPix[16] = Voxel[l][r+1][c];
   MaskPix[17] = Voxel[l][r+1][c+1];
   MaskPix[18] = Voxel[l+1][r-1][c-1];
   MaskPix[19] = Voxel[l+1][r-1][c];
   MaskPix[20] = Voxel[l+1][r+1][c+1];
   MaskPix[21] = Voxel[l+1][r][c-1];
   MaskPix[22] = Voxel[l+1][r][c];
   MaskPix[23] = Voxel[l+1][r][c+1];
   MaskPix[24] = Voxel[l+1][r+1][c-1];
   MaskPix[25] = Voxel[l+1][r+1][c];
   MaskPix[26] = Voxel[l+1][r+1][c+1];

  }

// Function: MinByIndex
int DistanceTransform3D :: MinByIndex(int from, int to)
  {
   int i;
   int temp;
```

```
   int Idx;

   temp = MaskPix[from];
   Idx = from;
   for (i = from; i <= to ; i ++)
     {
      if (temp > MaskPix[i])
        {
         temp = MaskPix[i];
         Idx = i;
        }
     }
   return Idx;
  }

#define min(a, b) (((a) < (b)) ? (a) : (b))

// Function Minimum of 5 values
int DistanceTransform3D :: Min5(int a, int b, int c, int d, int e)
  {
   int mn;

   mn = min(a, b);
   mn = min(mn, c);
   mn = min(mn, d);
   mn = min(mn, e);
   return mn;
  }

// Procedure: DistancePassOne
void DistanceTransform3D :: DistancePassOne(Voxel3D Voxel)
  {
   int r;
   int c;
   int l;
   int k;

   for (l = 1; l < maxlevel-1; l ++)
     {
      for (r = 1; r < maxrow-1; r ++)
        {
         for (c = 1; c < maxcol-1; c ++)
           {
            GetUpperMask(l, r, c, Voxel, MaskPix);
            for (k = 0; k < 13; k ++)
               {
                if ((k == 0) || (k == 2) ||
                    (k == 6) || (k == 8))
                   MaskPix[k] = MaskPix[k] + 5;

                if ((k == 1) || (k == 3) ||
                    (k == 5) || (k == 7) ||
                    (k == 9) || (k == 11))
                   MaskPix[k] = MaskPix[k] + 4;

                if ((k == 4) || (k == 10) || (k == 12))
                   MaskPix[k] = MaskPix[k] + 3;
                }

            if (MaskPix[13] != 30000)
```

```
                MaskPix[13] = 0;

            Voxel[l][r][c] = MaskPix[MinByIndex(0, 13)];
            //printf("\r");
            //cout << "level: " << l << " row: " << r << " col: " << c;
            //cout << "  Voxel[l][r][c]: " << Voxel[l][r][c];
          }
        }
      }
  }

// Procedure: DistancePassTwo
void DistanceTransform3D :: DistancePassTwo(Voxel3D Voxel)
  {
   int r;
   int c;
   int l;
   int k;

   for (l = maxlevel - 2; l > 0; l --)
     {
      for (r = maxrow - 2; r > 0; r --)
        {
         for (c = maxcol - 2; c > 0; c --)
            {
             GetLowerMask(l, r, c, Voxel, MaskPix);
             for (k = 26; k > 13; k --)
               {
                 if ((k == 18) || (k == 20) ||
                     (k == 24) || (k == 26))
                    MaskPix[k] = MaskPix[k] + 5;

                 if ((k == 19) || (k == 21) ||
                     (k == 23) || (k == 25) ||
                     (k == 15) || (k == 17))
                    MaskPix[k] = MaskPix[k] + 4;

                 if ((k == 14) || (k == 16) || (k == 22))
                    MaskPix[k] = MaskPix[k] + 3;
               }
             Voxel[l][r][c] = MaskPix[MinByIndex(13, 26)];
             }
        }
     }
   }


// Procedure: ForwardDistance
void DistanceTransform3D :: ForwardDistance()
  {
   cout << "3D Forward Distance ..." << endl;
   ReadVoxelImage(Voxel);
   SetBackground(Voxel, 30000, 0);
   DistancePassOne(Voxel);
  }

// Procedure: BackwardDistance
void DistanceTransform3D :: BackwardDistance()
  {
   cout << "3D Backward Distance ..." << endl;
   DistancePassTwo(Voxel);
```

```
  }


// Procedure: Title
void DistanceTransform3D :: Title()
  {
   cout << endl;
     cout  <<  "----------------Object-Oriented  3D  Distance  Transformation
--------------" << endl;
   cout << "------------ By: Alias Abdul-Rahman, November, 1998 ------------"
<< endl;
   cout << "   Input/Output : 3D array ILWIS look-like file format. " << endl;
   cout << endl;
  }
```

```
// Object-Oriented 3D DistanceTransform Program
// File: TDistanceTransform3D.h

#include "TDistanceTransform3D.h"

// Main program
void main()
  {
   TDistanceTransform3D ThreeDDT;     // ThreeDDT is an object
   ThreeDDT.Title();
   ThreeDDT.GetVPDInputFile();
   ThreeDDT.ForwardDistance();
   ThreeDDT.BackwardDistance();
   ThreeDDT.WriteVPDOutputFile();
   ThreeDDT.WriteVPIOutputFile();
   ThreeDDT.PressAnyKey();
  }

// End of main program

// Definitions sections follows:
// The constructor
TDistanceTransform3D :: TDistanceTransform3D()
  {
   GetVPIInputFile(VPI);

   // Allocate memory dynamically
   Voxel = new Voxel2D[maxlevel];
   for (l = 0; l < maxlevel; l ++)
     {
      Voxel[l] = new VoxelRow[maxrow];
      for (r = 0; r < maxrow; r ++)
        {
         Voxel[l][r] = new Image[maxcol];
        }
     }
  }

// The destructor
TDistanceTransform3D :: ~TDistanceTransform3D()
  {
   // Deallocate memory
   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         delete [] Voxel[l][r];
        }
     }
   delete [] Voxel;
  }

// Procedure: PressAnyKey
void TDistanceTransform3D :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
```

```
   cout << endl;
  }


// Procedure: GetVoxelInputFile
void TDistanceTransform3D :: GetVPDInputFile()
  {
   char vpdifilename [] = "c:\\data\\pnt5.vpd";

   VPDifile = fopen(vpdifilename, "rb");
   if (VPDifile == NULL)
     {
      cerr << "Could not open VPD file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The VPD file is opened" << endl;
     }
  }

// Procedure: GetVPIInputFile
void TDistanceTransform3D :: GetVPIInputFile(VpiType& VPI)
  {
   char vpiifilename[] = "c:\\data\\pnt5.vpi";
   VPIifile = fopen(vpiifilename, "rb");
   if (VPIifile == NULL)
     {
      cerr << "Could not open VPI file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "Information of  " << vpiifilename << endl;
      fread(&VPI, sizeof(VpiType), 1, VPIifile);
      int size = sizeof(VpiType);

      maxrow   = VPI.Nscanlines;
      maxcol   = VPI.Npixels;
      maxlevel = VPI.Nlevels;

      cout << "VPI file size       : " << size << endl;
      cout << endl;
      cout << "Number of lines     : " << VPI.Nscanlines << endl;
      cout << "Number of columns   : " << VPI.Npixels << endl;
      cout << "Number of levels    : " << VPI.Nlevels << endl;
      cout << "Minimum value       : " << VPI.Pvmin << endl;
      cout << "Maximum value       : " << VPI.Pvmax << endl;
      cout << "Map type            : " << VPI.Maptype << endl;
      cout << "Patched             : " << VPI.Patch << endl;
      cout << "Scale (power of 10) : " << VPI.Scale << endl;
      cout << "Coordinate type     : " << VPI.Crdtype << endl;
      cout << "Transformation parameters :" << endl;
      cout << "   a11  : " << VPI.a11 << endl;
      cout << "   a12  : " << VPI.a12 << endl;
      cout << "   a13  : " << VPI.a31 << endl;
      cout << "   a21  : " << VPI.a21 << endl;
      cout << "   a22  : " << VPI.a22 << endl;
      cout << "   a23  : " << VPI.a23 << endl;
```

```
      cout << "   a31   : " << VPI.a31 << endl;
      cout << "   a32   : " << VPI.a32 << endl;
      cout << "   a33   : " << VPI.a33 << endl;
      cout << endl;
      cout << "Translation vectors : " << endl;
      cout << "   b1    : " << VPI.b1 << endl;
      cout << "   b2    : " << VPI.b2 << endl;
      cout << "   b3    : " << VPI.b3 << endl;
      fclose(VPIifile);
    }
  }

// Procedure: ReadVoxelImage
void TDistanceTransform3D :: ReadVoxelImage(Voxel3D& Voxel)
  {
   int r;
   int c;
   int l;

   cout << endl;
   cout << "Reading voxels data ..., wait !" << endl;
   cout << endl;

   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         for (c = 0; c < maxcol; c ++)
           {
            fread(&Voxel[l][r][c], sizeof(short), 1, VPDifile);
           }
        }
     }
   cout << endl;
   fclose(VPDifile);
  }

// Procedure: PrintPixel
void TDistanceTransform3D :: PrintVoxel(Voxel3D Voxel)
  {
   int r;
   int c;
   int l;

   for (r = 0; r < maxrow; r ++)
     {
      for (c = 0; c < maxcol; c ++)
        {
         for (l = 0; l < maxlevel; l ++)
           {
            cout << "Voxel value at [row][col][level]: [" << r << "][" << c
                 << "][" << l << "]: " << Voxel[r][c][l] << "\r";
           }
        }
     }
  }


// Procedure: WriteVoxelOutputFile
void TDistanceTransform3D :: WriteVPDOutputFile()
  {
```

```
    int r;
    int l;
    int c;

    char vpdofilename[] = "c:\\data\\pnt5od.vpd";
    VPDofile = fopen(vpdofilename, "w+b");
    for (l = 0; l < maxlevel; l ++)
      {
       for (r = 0; r < maxrow; r ++)
         {
          for (c = 0; c < maxcol; c ++)
            {
             fwrite(&Voxel[l][r][c], sizeof(short), 1, VPDofile);
            }
         }
      }
    fclose(VPDofile);
  }

// Procedure: WriteVPIFile
void TDistanceTransform3D :: WriteVPIOutputFile()
  {
    FILE* VPIofile;
    char vpifilename[] = "c:\\data\\pnt5od.vpi";
    VPIofile = fopen(vpifilename, "wb");

    // assign the value for the VPI file
    VPIOut.Nscanlines = VPI.Nscanlines;
    VPIOut.Npixels = VPI.Npixels;
    VPIOut.Nlevels = VPI.Nlevels;
    VPIOut.Pvmin = 0;
    VPIOut.Pvmax = 6000;
    VPIOut.Maptype = 2;
    VPIOut.Patch = 0;
    VPIOut.Scale = 0;
    VPIOut.Crdtype = 1;
    VPIOut.a11 = VPI.a11;
    VPIOut.a12 = 0;
    VPIOut.a13 = 0;
    VPIOut.a21 = 0;
    VPIOut.a22 = VPI.a22;
    VPIOut.a23 = 0;
    VPIOut.a31 = 0;
    VPIOut.a32 = 0;
    VPIOut.a33 = VPI.a33;
    VPIOut.b1 = 1;
    VPIOut.b2 = 1;
    VPIOut.b3 = maxlevel + 1;

    // write out the above values to the VPI file
    fwrite(&VPIOut, sizeof(VpiType), 1, VPIofile);
    fclose(VPIofile);
  }


// Procedure: SetBackground
void TDistanceTransform3D :: SetBackground(Voxel3D Voxel, int Bg, int Fg)
  {
    int row;
    int col;
    int l;
```

```
   for (l = 0; l < maxlevel; l ++)
     {
      for (row = 0; row < maxrow; row ++)
        {
         for (col = 0; col < maxcol; col ++)
           {
            if (Voxel[l][row][col] == 0)
                Voxel[l][row][col] = Bg;
            else
            if (Fg > 0)
                Voxel[l][row][col] = Fg;
            }

        }
     }
  }

// Procedure: GetUpperMask
void TDistanceTransform3D :: GetUpperMask(int l, int r, int c, Voxel3D Voxel,
Mask& MaskPix)
  {
   MaskPix[0]  = Voxel[l-1][r-1][c-1];
   MaskPix[1]  = Voxel[l-1][r-1][c];
   MaskPix[2]  = Voxel[l-1][r-1][c+1];
   MaskPix[3]  = Voxel[l-1][r][c-1];
   MaskPix[4]  = Voxel[l-1][r][c];
   MaskPix[5]  = Voxel[l-1][r][c+1];
   MaskPix[6]  = Voxel[l-1][r+1][c-1];
   MaskPix[7]  = Voxel[l-1][r+1][c];
   MaskPix[8]  = Voxel[l-1][r+1][c+1];
   MaskPix[9]  = Voxel[l][r-1][c-1];
   MaskPix[10] = Voxel[l][r-1][c];
   MaskPix[11] = Voxel[l][r-1][c+1];
   MaskPix[12] = Voxel[l][r][c-1];
   MaskPix[13] = Voxel[l][r][c];

  }

// Procedure: GetLowerMask
void TDistanceTransform3D :: GetLowerMask(int l, int r, int c, Voxel3D Voxel,
Mask& MaskPix)
  {
   MaskPix[13] = Voxel[l][r][c];
   MaskPix[14] = Voxel[l][r][c+1];
   MaskPix[15] = Voxel[l][r+1][c-1];
   MaskPix[16] = Voxel[l][r+1][c];
   MaskPix[17] = Voxel[l][r+1][c+1];
   MaskPix[18] = Voxel[l+1][r-1][c-1];
   MaskPix[19] = Voxel[l+1][r-1][c];
   MaskPix[20] = Voxel[l+1][r+1][c+1];
   MaskPix[21] = Voxel[l+1][r][c-1];
   MaskPix[22] = Voxel[l+1][r][c];
   MaskPix[23] = Voxel[l+1][r][c+1];
   MaskPix[24] = Voxel[l+1][r+1][c-1];
   MaskPix[25] = Voxel[l+1][r+1][c];
   MaskPix[26] = Voxel[l+1][r+1][c+1];

  }

// Function: MinByIndex
int TDistanceTransform3D :: MinByIndex(int from, int to)
```

```
  {
   int i;
   int temp;
   int Idx;

   temp = MaskPix[from];
   Idx = from;
   for (i = from; i <= to ; i ++)
     {
      if (temp > MaskPix[i])
        {
         temp = MaskPix[i];
         Idx = i;
        }
     }
   return Idx;
  }

#define min(a, b) (((a) < (b)) ? (a) : (b))

// Function Minimum of 5 values
int TDistanceTransform3D :: Min5(int a, int b, int c, int d, int e)
  {
   int mn;

   mn = min(a, b);
   mn = min(mn, c);
   mn = min(mn, d);
   mn = min(mn, e);
   return mn;
  }

// Procedure: DistancePassOne
void TDistanceTransform3D :: DistancePassOne(Voxel3D Voxel)
  {
   int r;
   int c;
   int l;
   int k;

   for (l = 1; l < maxlevel-1; l ++)
     {
      for (r = 1; r < maxrow-1; r ++)
        {
         for (c = 1; c < maxcol-1; c ++)
           {
            GetUpperMask(l, r, c, Voxel, MaskPix);
            for (k = 0; k < 13; k ++)
               {
                if ((k == 0) || (k == 2) ||
                    (k == 6) || (k == 8))
                   MaskPix[k] = MaskPix[k] + 5;

                if ((k == 1) || (k == 3) ||
                    (k == 5) || (k == 7) ||
                    (k == 9) || (k == 11))
                   MaskPix[k] = MaskPix[k] + 4;

                if ((k == 4) || (k == 10) || (k == 12))
                   MaskPix[k] = MaskPix[k] + 3;
```

```
                }

            if (MaskPix[13] != 30000)
                MaskPix[13] = 0;

            Voxel[l][r][c] = MaskPix[MinByIndex(0, 13)];
            //printf("\r");
            //cout << "level: " << l << " row: " << r << " col: " << c;
            //cout << "  Voxel[l][r][c]: " << Voxel[l][r][c];
          }
        }
      }
  }

// Procedure: DistancePassTwo
void TDistanceTransform3D :: DistancePassTwo(Voxel3D Voxel)
  {
   int r;
   int c;
   int l;
   int k;

   for (l = maxlevel - 2; l > 0; l --)
     {
      for (r = maxrow - 2; r > 0; r --)
        {
         for (c = maxcol - 2; c > 0; c --)
            {
             GetLowerMask(l, r, c, Voxel, MaskPix);
             for (k = 26; k > 13; k --)
               {
                 if ((k == 18) || (k == 20) ||
                     (k == 24) || (k == 26))
                    MaskPix[k] = MaskPix[k] + 5;

                 if ((k == 19) || (k == 21) ||
                     (k == 23) || (k == 25) ||
                     (k == 15) || (k == 17))
                    MaskPix[k] = MaskPix[k] + 4;

                 if ((k == 14) || (k == 16) || (k == 22))
                    MaskPix[k] = MaskPix[k] + 3;
               }
             Voxel[l][r][c] = MaskPix[MinByIndex(13, 26)];
             }
        }
     }
   }


// Procedure: ForwardDistance
void TDistanceTransform3D :: ForwardDistance()
  {
   cout << "3D Forward Distance ..." << endl;
   ReadVoxelImage(Voxel);
   SetBackground(Voxel, 30000, 0);
   DistancePassOne(Voxel);
  }

// Procedure: BackwardDistance
void TDistanceTransform3D :: BackwardDistance()
```

```
  {
   cout << "3D Backward Distance ..." << endl;
   DistancePassTwo(Voxel);
  }


// Procedure: Title
void TDistanceTransform3D :: Title()
  {
   cout << endl;
      cout  <<  "---------------Object-Oriented  3D  Distance  Transformation
--------------" << endl;
   cout << "----------- By: Alias Abdul-Rahman, November, 1998 ------------"
<< endl;
   cout << "   Input/Output : 3D array ILWIS look-like file format. " << endl;
   cout << endl;
  }
```

3D Voronoi Tessellations Code

The following gives the complete codes for the 3D Voronoi tessellations program.

```
//*************************************************************************//
//          Object-Oriented 3D Distance Transformation (DT) and
//
//                Voronoi Tessellation program for 3D TIN
//
//
//
//              Input   :  3D array format
//
//              Output  :  3D array ILWIS look-like file format
//
//                                 (i.e. .VPD, and .VPI)
//
//
//
//*************************************************************************//

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <io.h>
#include <iomanip.h>
#include <dos.h>
#include <malloc.h>
#include <conio.h>
#include <limits.h>
#include <stddef.h>

#include "TDistanceTransform3D.h"

// Constant section

#define masksize 27                 // Mask size

class TVoronoi3D : public TDistanceTransform3D
  {
   public:
      TVoronoi3D();             // constructor
      ~TVoronoi3D();            // destructor

     typedef short Mask[masksize];

     // Global variables
     Mask MaskPixVor;

     FILE* VPDifile;
     FILE* VPDodfile;
```

```
    FILE* VPDovfile;
    FILE* VPIifile;
    FILE* VPIodfile;
    FILE* VPIovfile;

    // variable definition
    Voxel3D Voxel, VoxelVor;          // variable for voxels

    // Functions Prototype;
    void GetVPDInputFile();
    void GetVPIInputFile(VpiType& VPI);
    void ReadVoxelImage(Voxel3D&);
    void CopyVoxel(Voxel3D, Voxel3D&);
    void PrintVoxel(Voxel3D);
    void WriteVPDOutputFileDist();
    void WriteVPIOutputFileDist();
    void WriteVPDOutputFileVoronoi();
    void WriteVPIOutputFileVoronoi();
    void SetBackground(Voxel3D, int, int);
    void GetUpperMaskDist(int, int, int, Voxel3D, Mask&);
    void GetLowerMaskDist(int, int, int, Voxel3D, Mask&);
    void GetUpperMaskVoronoi(int, int, int, Voxel3D, Mask&);
    void GetLowerMaskVoronoi(int, int, int, Voxel3D, Mask&);
    int MinByIndex(int, int);
    int Min5(int, int, int, int, int);
    void ForwardPass(Voxel3D, Voxel3D);
    void BackwardPass(Voxel3D, Voxel3D);
    void ForwardVoronoi();
    void BackwardVoronoi();
    void PressAnyKey();
    void Title();
};
```

```
// Object-Oriented 3D Voronoi Tessellation Program
// File: TVor3D.cpp

#include "TVor3D.h"

// Main program
void main()
  {
   TVoronoi3D Vor3D;
   Vor3D.Title();
   TVoronoi3D();              // constructor
   Vor3D.GetVPDInputFile();
   Vor3D.ForwardVoronoi();
   Vor3D.BackwardVoronoi();
   Vor3D.WriteVPDOutputFileDist();
   Vor3D.WriteVPIOutputFileDist();
   Vor3D.WriteVPDOutputFileVoronoi();
   Vor3D.WriteVPIOutputFileVoronoi();
   Vor3D.PressAnyKey();
  }

// End of main program

// Definitions sections
// The constructor
TVoronoi3D :: TVoronoi3D()
  {
   GetVPIInputFile(VPI);

   // Allocate memory dynamically for DT
   Voxel = new Voxel2D[maxlevel];
   for (l = 0; l < maxlevel; l ++)
     {
      Voxel[l] = new VoxelRow[maxrow];
      for (r = 0; r < maxrow; r ++)
        {
         Voxel[l][r] = new Image[maxcol];
        }
     }

   // Allocate memory dynamically for Voronoi
   VoxelVor = new Voxel2D[maxlevel];
   for (l = 0; l < maxlevel; l ++)
     {
      VoxelVor[l] = new VoxelRow[maxrow];
      for (r = 0; r < maxrow; r ++)
        {
         VoxelVor[l][r] = new Image[maxcol];
        }
     }
  }

// The destructor
TVoronoi3D :: ~TVoronoi3D()
  {
   // Deallocate memory for DT
   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         delete [] Voxel[l][r];
```

```
        }
      }
    delete [] Voxel;

    // Deallocate memory for Voronoi
    for (l = 0; l < maxlevel; l ++)
      {
       for (r = 0; r < maxrow; r ++)
         {
          delete [] VoxelVor[l][r];
         }
      }
    delete [] VoxelVor;
  }

// Procedure: PressAnyKey
void TVoronoi3D :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }


// Procedure: GetVoxelInputFile
void TVoronoi3D :: GetVPDInputFile()
  {
   char vpdifilename [] = "c:\\data\\ras3d.vpd";

   VPDifile = fopen(vpdifilename, "rb");
   if (VPDifile == NULL)
     {
      cerr << "Could not open VPD file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The VPD file is opened" << endl;
     }
  }

// Procedure: GetVPIInputFile
void TVoronoi3D :: GetVPIInputFile(VpiType& VPI)
  {
   char vpiifilename[] = "c:\\data\\ras3d.vpi";
   VPIifile = fopen(vpiifilename, "rb");
   if (VPIifile == NULL)
     {
      cerr << "Could not open VPI file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "Information of  " << vpiifilename << endl;
      fread(&VPI, sizeof(VpiType), 1, VPIifile);
      int size = sizeof(VpiType);

      maxrow   = VPI.Nscanlines;
```

```
      maxcol   = VPI.Npixels;
      maxlevel = VPI.Nlevels;

      cout << "VPI file size       : " << size << endl;
      cout << endl;
      cout << "Number of lines     : " << VPI.Nscanlines << endl;
      cout << "Number of columns   : " << VPI.Npixels << endl;
      cout << "Number of levels    : " << VPI.Nlevels << endl;
      cout << "Minimum value       : " << VPI.Pvmin << endl;
      cout << "Maximum value       : " << VPI.Pvmax << endl;
      cout << "Map type            : " << VPI.Maptype << endl;
      cout << "Patched             : " << VPI.Patch << endl;
      cout << "Scale (power of 10) : " << VPI.Scale << endl;
      cout << "Coordinate type     : " << VPI.Crdtype << endl;
      cout << "Transformation parameters :" << endl;
      cout << "   a11  : " << VPI.a11 << endl;
      cout << "   a12  : " << VPI.a12 << endl;
      cout << "   a13  : " << VPI.a31 << endl;
      cout << "   a21  : " << VPI.a21 << endl;
      cout << "   a22  : " << VPI.a22 << endl;
      cout << "   a23  : " << VPI.a23 << endl;
      cout << "   a31  : " << VPI.a31 << endl;
      cout << "   a32  : " << VPI.a32 << endl;
      cout << "   a33  : " << VPI.a33 << endl;
      cout << endl;
      cout << "Translation vectors : " << endl;
      cout << "   b1   : " << VPI.b1 << endl;
      cout << "   b2   : " << VPI.b2 << endl;
      cout << "   b3   : " << VPI.b3 << endl;
      fclose(VPIifile);
     }
  }

// Procedure: ReadVoxelImage
void TVoronoi3D :: ReadVoxelImage(Voxel3D& Voxel)
  {
   int r;
   int c;
   int l;

   cout << endl;
   cout << "Reading voxels data ..., wait ! " << endl;
   cout << endl;

   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         for (c = 0; c < maxcol; c ++)
           {
            fread(&Voxel[l][r][c], sizeof(short), 1, VPDifile);
           }
        }
     }
   fclose(VPDifile);
   cout << endl;
  }

// Procedure: CopyVoxel
void TVoronoi3D :: CopyVoxel(Voxel3D VoxA, Voxel3D& VoxB)
  {
```

```
    int i, j, k;
    for (i = 0; i < maxlevel; i ++)
      {
       for (j = 0; j < maxrow; j ++)
         {
          for (k = 0; k < maxcol; k ++)
            {
             VoxB[i][j][k] = VoxA[i][j][k];
            }
         }
      }
  }

// Procedure: PrintPixel
void TVoronoi3D :: PrintVoxel(Voxel3D Voxel)
  {
   int r;
   int c;
   int l;

   for (r = 0; r < maxrow; r ++)
     {
      for (c = 0; c < maxcol; c ++)
        {
         for (l = 0; l < maxlevel; l ++)
           {
            cout << "Voxel value at [row][col][level]: [" << r << "][" << c
                 << "][" << l << "]: " << Voxel[r][c][l] << "\r";
           }
        }
     }
  }


// Procedure: WriteVoxelOutputFile
void TVoronoi3D :: WriteVPDOutputFileDist()
  {
   int r;
   int l;
   int c;

   char vpdodfilename[] = "c:\\data\\tvoxod.vpd";
   VPDodfile = fopen(vpdodfilename, "wb");
   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         for (c = 0; c < maxcol; c ++)
           {
            fwrite(&Voxel[l][r][c], sizeof(short), 1, VPDodfile);
           }
        }
     }
   fclose(VPDodfile);
  }

// Procedure: WriteVPIFile
void TVoronoi3D :: WriteVPIOutputFileDist()
  {
   FILE* VPIofile;
```

```
    char vpifilename[] = "c:\\data\\tvoxod.vpi";
    VPIofile = fopen(vpifilename, "wb");

    // assign the value for the VPI file
    VPIOut.Nscanlines = VPI.Nscanlines;
    VPIOut.Npixels = VPI.Npixels;
    VPIOut.Nlevels = VPI.Nlevels;
    VPIOut.Pvmin = 0;
    VPIOut.Pvmax = 200;
    VPIOut.Maptype = 2;
    VPIOut.Patch = 0;
    VPIOut.Scale = 0;
    VPIOut.Crdtype = 1;
    VPIOut.a11 = VPI.a11;
    VPIOut.a12 = 0;
    VPIOut.a13 = 0;
    VPIOut.a21 = 0;
    VPIOut.a22 = VPI.a22;
    VPIOut.a23 = 0;
    VPIOut.a31 = 0;
    VPIOut.a32 = 0;
    VPIOut.a33 = VPI.a33;
    VPIOut.b1 = 1;
    VPIOut.b2 = 1;
    VPIOut.b3 = maxlevel + 1;

    // write out the above values to the VPI file
    fwrite(&VPIOut, sizeof(VpiType), 1, VPIofile);
    fclose(VPIofile);
  }

// Procedure: WriteVPDOutputFileVoronoi
void TVoronoi3D :: WriteVPDOutputFileVoronoi()
  {
    int r;
    int l;
    int c;

    char vpdovfilename[] = "c:\\data\\tvoxov.vpd";
    VPDovfile = fopen(vpdovfilename, "wb");
    for (l = 0; l < maxlevel; l ++)
      {
       for (r = 0; r < maxrow; r ++)
         {
          for (c = 0; c < maxcol; c ++)
            {
             fwrite(&VoxelVor[l][r][c], sizeof(short), 1, VPDovfile);
            }
         }
      }
    fclose(VPDovfile);
  }

// Procedure: WriteVPIOutputFileVoronoi
void TVoronoi3D :: WriteVPIOutputFileVoronoi()
  {
    FILE* VPIovfile;
    char vpiovfilename[] = "c:\\data\\tvoxov.vpi";
    VPIovfile = fopen(vpiovfilename, "wb");

    // assign the value for the VPI file
```

```
    VPIOut.Nscanlines = VPI.Nscanlines;
    VPIOut.Npixels = VPI.Npixels;
    VPIOut.Nlevels = VPI.Nlevels;
    VPIOut.Pvmin = 0;
    VPIOut.Pvmax = 200;
    VPIOut.Maptype = 2;
    VPIOut.Patch = 0;
    VPIOut.Scale = 0;
    VPIOut.Crdtype = 1;
    VPIOut.a11 = VPI.a11;
    VPIOut.a12 = 0;
    VPIOut.a13 = 0;
    VPIOut.a21 = 0;
    VPIOut.a22 = VPI.a22;
    VPIOut.a23 = 0;
    VPIOut.a31 = 0;
    VPIOut.a32 = 0;
    VPIOut.a33 = VPI.a33;
    VPIOut.b1 = 1;
    VPIOut.b2 = 1;
    VPIOut.b3 = maxlevel + 1;

    // write out the above values to the VPI file
    fwrite(&VPIOut, sizeof(VpiType), 1, VPIovfile);
    fclose(VPIovfile);
  }


// Procedure: SetBackground
void TVoronoi3D :: SetBackground(Voxel3D Voxel, int Bg, int Fg)
  {
   int row;
   int col;
   int l;

   for (l = 0; l < maxlevel; l ++)
     {
      for (row = 0; row < maxrow; row ++)
        {
         for (col = 0; col < maxcol; col ++)
           {
            if (Voxel[l][row][col] == 0)
                Voxel[l][row][col] = Bg;
            else
            if (Fg > 0)
                Voxel[l][row][col] = Fg;
            }

        }
     }
  }

// Procedure: GetUpperMaskDist
void TVoronoi3D :: GetUpperMaskDist(int l, int r, int c, Voxel3D Voxel, Mask&
MaskPix)
  {
   MaskPix[0]  = Voxel[l-1][r-1][c-1];
   MaskPix[1]  = Voxel[l-1][r-1][c];
   MaskPix[2]  = Voxel[l-1][r-1][c+1];
   MaskPix[3]  = Voxel[l-1][r][c-1];
   MaskPix[4]  = Voxel[l-1][r][c];
```

```
   MaskPix[5]  = Voxel[l-1][r][c+1];
   MaskPix[6]  = Voxel[l-1][r+1][c-1];
   MaskPix[7]  = Voxel[l-1][r+1][c];
   MaskPix[8]  = Voxel[l-1][r+1][c+1];
   MaskPix[9]  = Voxel[l][r-1][c-1];
   MaskPix[10] = Voxel[l][r-1][c];
   MaskPix[11] = Voxel[l][r-1][c+1];
   MaskPix[12] = Voxel[l][r][c-1];
   MaskPix[13] = Voxel[l][r][c];

  }

// Procedure: GetLowerMaskDist
void TVoronoi3D :: GetLowerMaskDist(int l, int r, int c, Voxel3D Voxel, Mask&
MaskPix)
  {
   MaskPix[13] = Voxel[l][r][c];
   MaskPix[14] = Voxel[l][r][c+1];
   MaskPix[15] = Voxel[l][r+1][c-1];
   MaskPix[16] = Voxel[l][r+1][c];
   MaskPix[17] = Voxel[l][r+1][c+1];
   MaskPix[18] = Voxel[l+1][r-1][c-1];
   MaskPix[19] = Voxel[l+1][r-1][c];
   MaskPix[20] = Voxel[l+1][r+1][c+1];
   MaskPix[21] = Voxel[l+1][r][c-1];
   MaskPix[22] = Voxel[l+1][r][c];
   MaskPix[23] = Voxel[l+1][r][c+1];
   MaskPix[24] = Voxel[l+1][r+1][c-1];
   MaskPix[25] = Voxel[l+1][r+1][c];
   MaskPix[26] = Voxel[l+1][r+1][c+1];

  }

// Procedure: GetUpperMaskVoronoi
void TVoronoi3D :: GetUpperMaskVoronoi(int l, int r, int c, Voxel3D VoxelVor,
Mask& MaskPixVor)
  {
   MaskPixVor[0]  = VoxelVor[l-1][r-1][c-1];
   MaskPixVor[1]  = VoxelVor[l-1][r-1][c];
   MaskPixVor[2]  = VoxelVor[l-1][r-1][c+1];
   MaskPixVor[3]  = VoxelVor[l-1][r][c-1];
   MaskPixVor[4]  = VoxelVor[l-1][r][c];
   MaskPixVor[5]  = VoxelVor[l-1][r][c+1];
   MaskPixVor[6]  = VoxelVor[l-1][r+1][c-1];
   MaskPixVor[7]  = VoxelVor[l-1][r+1][c];
   MaskPixVor[8]  = VoxelVor[l-1][r+1][c+1];
   MaskPixVor[9]  = VoxelVor[l][r-1][c-1];
   MaskPixVor[10] = VoxelVor[l][r-1][c];
   MaskPixVor[11] = VoxelVor[l][r-1][c+1];
   MaskPixVor[12] = VoxelVor[l][r][c-1];
   MaskPixVor[13] = VoxelVor[l][r][c];
  }

// Procedure: GetLowerMaskVoronoi
void TVoronoi3D :: GetLowerMaskVoronoi(int l, int r, int c, Voxel3D VoxelVor,
Mask& MaskPixVor)
  {
   MaskPixVor[13] = VoxelVor[l][r][c];
   MaskPixVor[14] = VoxelVor[l][r][c+1];
   MaskPixVor[15] = VoxelVor[l][r+1][c-1];
   MaskPixVor[16] = VoxelVor[l][r+1][c];
```

```
    MaskPixVor[17] = VoxelVor[l][r+1][c+1];
    MaskPixVor[18] = VoxelVor[l+1][r-1][c-1];
    MaskPixVor[19] = VoxelVor[l+1][r-1][c];
    MaskPixVor[20] = VoxelVor[l+1][r+1][c+1];
    MaskPixVor[21] = VoxelVor[l+1][r][c-1];
    MaskPixVor[22] = VoxelVor[l+1][r][c];
    MaskPixVor[23] = VoxelVor[l+1][r][c+1];
    MaskPixVor[24] = VoxelVor[l+1][r+1][c-1];
    MaskPixVor[25] = VoxelVor[l+1][r+1][c];
    MaskPixVor[26] = VoxelVor[l+1][r+1][c+1];
  }

// Function: MinByIndex
int TVoronoi3D :: MinByIndex(int from, int to)
  {
   int i;
   int temp;
   int Idx;

   temp = MaskPix[from];
   Idx = from;
   for (i = from; i <= to ; i ++)
     {
      if (temp > MaskPix[i])
        {
         temp = MaskPix[i];
         Idx = i;
        }
     }
   return Idx;
  }

#define min(a, b) (((a) < (b)) ? (a) : (b))

// Function Minimum of 5 values
int TVoronoi3D :: Min5(int a, int b, int c, int d, int e)
  {
   int mn;

   mn = min(a, b);
   mn = min(mn, c);
   mn = min(mn, d);
   mn = min(mn, e);
   return mn;
  }

// Procedure: ForwardPass
void TVoronoi3D :: ForwardPass(Voxel3D Voxel, Voxel3D VoxelVor)
  {
   int r;
   int c;
   int l;
   int k;

   for (l = 1; l < maxlevel-1; l ++)
     {
      for (r = 1; r < maxrow-1; r ++)
        {
         for (c = 1; c < maxcol-1; c ++)
           {
            GetUpperMaskDist(l, r, c, Voxel, MaskPix);
```

```
            GetUpperMaskVoronoi(l, r, c, VoxelVor, MaskPixVor);
            for (k = 0; k < 13; k ++)
               {
                if ((k == 0) || (k == 2) ||
                    (k == 6) || (k == 8))
                   MaskPix[k] = MaskPix[k] + 5;

                if ((k == 1) || (k == 3) ||
                    (k == 5) || (k == 7) ||
                    (k == 9) || (k == 11))
                   MaskPix[k] = MaskPix[k] + 4;

                if ((k == 4) || (k == 10) || (k == 12))
                   MaskPix[k] = MaskPix[k] + 3;
                }

            if (MaskPix[13] != 30000)
                MaskPix[13] = 0;

            Voxel[l][r][c] = MaskPix[MinByIndex(0, 13)];
            VoxelVor[l][r][c] = MaskPixVor[MinByIndex(0, 13)];
           }
        }
     }
  }

// Procedure: BackwardPass
void TVoronoi3D :: BackwardPass(Voxel3D Voxel, Voxel3D VoxelVor)
  {
   int r;
   int c;
   int l;
   int k;

   for (l = maxlevel - 2; l > 0; l --)
     {
      for (r = maxrow - 2; r > 0; r --)
        {
         for (c = maxcol - 2; c > 0; c --)
            {
             GetLowerMaskDist(l, r, c, Voxel, MaskPix);
             GetLowerMaskVoronoi(l, r, c, VoxelVor, MaskPixVor);
             for (k = 26; k > 13; k --)
               {
                if ((k == 18) || (k == 20) ||
                    (k == 24) || (k == 26))
                   MaskPix[k] = MaskPix[k] + 5;

                if ((k == 19) || (k == 21) ||
                    (k == 23) || (k == 25) ||
                    (k == 15) || (k == 17))
                   MaskPix[k] = MaskPix[k] + 4;

                if ((k == 14) || (k == 16) || (k == 22))
                   MaskPix[k] = MaskPix[k] + 3;
               }
             Voxel[l][r][c] = MaskPix[MinByIndex(13, 26)];
             VoxelVor[l][r][c] = MaskPixVor[MinByIndex(13, 26)];
             }
        }
     }
```

```
    }

// Procedure: ForwardVoronoi
void TVoronoi3D :: ForwardVoronoi()
  {
   cout << "D3 Forward pass ..." << endl;
   ReadVoxelImage(Voxel);
   CopyVoxel(Voxel, VoxelVor);
   SetBackground(Voxel, 30000, 0);
   ForwardPass(Voxel, VoxelVor);
  }

// Procedure: BackwardVoronoi
void TVoronoi3D :: BackwardVoronoi()
  {
   cout << "3D Backward pass ..." << endl;
   BackwardPass(Voxel, VoxelVor);
  }

// Procedure: Title
void TVoronoi3D :: Title()
  {
   cout << endl;
   cout << "----------------- Object-Oriented 3D DT and Voronoi Tessellation
--------------" << endl;
   cout << "----------- By: Alias Abdul-Rahman, November, 1998 ------------"
<< endl;
   cout << "   Input/Output : 3D array ILWIS look-like file format. " << endl;
   cout << endl;
  }
```

3D TIN Data Structuring Code

The following gives the complete code for the 3D TIN (or TEN) data structure generation program.

```
// ***************************************************************************
//
//                      Object-Oriented 3D Voronoi-to-3D TIN (TEN) program
//
//               Input  :  VPI and VPD file of 3D Voronoi program
//
//               Output :  3D TIN (TEN) table (Ascii file)
//
//                         TEN = Tetrahedron Network
//
// ***************************************************************************
//

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <io.h>
#include <iomanip.h>
#include <dos.h>
#include <alloc.h>
#include <malloc.h>
#include <conio.h>
#include <limits.h>
#include <stddef.h>

#include "TVor3D.h"

#define masksize 8           // 2 x 2 x 2 Mask size

class TTinGen3D : public TVoronoi3D
  {
   public:
        TTinGen3D();
        ~TTinGen3D();

        typedef short DataType;
        typedef struct VertexStruct
                  {
                   DataType N1;
                   DataType N2;
                   DataType N3;
                   DataType N4;
                  } TVertex;

        typedef struct TsNodeStruct
                  {
```

```
                short x;
                short y;
               } TsNode;

        typedef short Image;
        typedef Image* VoxelRow;
        typedef VoxelRow* Voxel2D;
        typedef Voxel2D* Voxel3D;
        typedef short Mask[8];

        // Global variables
        Mask MaskPix;
        Voxel3D Voxel;
        FILE* VPDfile;
        FILE* VPIfile;

        FILE* TENfile;
        int nTEN;
        TVertex Element;
        TsNode TsPnt;
        bool FoundTEN;

        // Function members
        void GetVPDfile();
        void ReadVoxelImage(Voxel3D&);
        void GetVPIfile(VpiStruct&);
        void Get3DTINfile();
        bool Less(DataType, DataType);
        bool Greater(DataType, DataType);
        void Swap(DataType&, DataType&);
        void NodeOrder(DataType&, DataType&, DataType&, DataType&);
        void AddTENtoFile(TVertex);
        void GetMaskVox(int, int, int, Voxel3D, Mask&);
        void GetSubImage(int, int, int, Voxel3D, Mask&);
        void DetectTEN(Mask);
        void ScanVoxels(Voxel3D);
        void Make3DTIN();
        void Title();
        void PressAnyKey();
};
```

```
// Object-Oriented 3D TIN Tessellation Program
// File: TVorTin3D.cpp

#include "TTinGen3D.h"


// main program
void main()
  {
   TTinGen3D VoxTEN;
   VoxTEN.Title();
   TTinGen3D();
   VoxTEN.GetVPDfile();
   VoxTEN.Get3DTINfile();
   VoxTEN.Make3DTIN();
   VoxTEN.PressAnyKey();
  }

// Definitions section
// The constructor
TTinGen3D :: TTinGen3D()
  {
   GetVPIfile(VPI);

   // Allocate memory dynamically
   Voxel = new Voxel2D[maxlevel];
   for (l = 0; l < maxlevel; l ++)
     {
      Voxel[l] = new VoxelRow[maxrow];
      for (r = 0; r < maxrow; r ++)
        {
         Voxel[l][r] = new Image[maxcol];
        }
     }
  }

TTinGen3D :: ~TTinGen3D()
  {
   // Deallocate memory
   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r ++)
        {
         delete [] Voxel[l][r];
        }
     }
   delete [] Voxel;
  }

// Procedure: PresAnyKey
void TTinGen3D :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

// Procedure: GetMPDFile
void TTinGen3D :: GetVPDfile()
  {
```

```
    char vpdfilename [] = "c:\\data\\pnt24v.vpd";
    VPDfile = fopen(vpdfilename, "rb");

    if (VPDfile == NULL)
      {
       cerr << "Could not open VPD file !" << endl;
       exit(1);
      }
    else
      {
       cout << endl;
       cout << "The VPD file is opened" << endl;
      }
  }

void TTinGen3D :: ReadVoxelImage(Voxel3D& Voxel)
  {
   int l, r, c;
   for (l = 0; l < maxlevel; l ++)
     {
      for (r = 0; r < maxrow; r++)
        {
         for (c = 0; c < maxcol; c ++)
           {
            fread(&Voxel[l][r][c], sizeof(short), 1, VPDfile);
           }
        }
     }
   cout << endl;
  fclose(VPDfile);
  }

void TTinGen3D :: GetVPIfile(VpiStruct& VPI)
  {
   char vpifilename [] = "c:\\data\\pnt24v.vpi";
   VPIfile = fopen(vpifilename, "rb");

   if (VPIfile == NULL)
     {
      cerr << "Could not open VPI file !" << endl;
      exit(1);
     }
   else
     {
      cout << "Information of  " << vpifilename << endl;
      fread(&VPI, sizeof(VpiType), 1, VPIfile);
      int size = sizeof(VpiType);

      maxrow = VPI.Nscanlines;
      maxcol = VPI.Npixels;
      maxlevel = VPI.Nlevels;

      cout << "MPI file size       : " << size << endl;
      cout << endl;
      cout << "Number of lines     : " << VPI.Nscanlines << endl;
      cout << "Number of columns   : " << VPI.Npixels << endl;
      cout << "Number of levels    : " << VPI.Nlevels << endl;
      cout << "Minimum value       : " << VPI.Pvmin << endl;
      cout << "Maximum value       : " << VPI.Pvmax << endl;
      cout << "Map type            : " << VPI.Maptype << endl;
      cout << "Patched             : " << VPI.Patch << endl;
```

```
      cout << "Scale (power of 10) : " << VPI.Scale << endl;
      cout << "Coordinate type     : " << VPI.Crdtype << endl;
      cout << "Transformation parameters :" << endl;
      cout << "   a11  : " << VPI.a11 << endl;
      cout << "   a12  : " << VPI.a12 << endl;
      cout << "   a13  : " << VPI.a13 << endl;
      cout << "   a21  : " << VPI.a21 << endl;
      cout << "   a22  : " << VPI.a22 << endl;
      cout << "   a23  : " << VPI.a23 << endl;
      cout << "   a31  : " << VPI.a31 << endl;
      cout << "   a32  : " << VPI.a32 << endl;
      cout << "   a33  : " << VPI.a33 << endl;
      cout << "Translation vectors: " << endl;
      cout << "   b1   : " << VPI.b1 << endl;
      cout << "   b2   : " << VPI.b2 << endl;
      cout << "   b3   : " << VPI.b3 << endl;
      fclose(VPIfile);
     }
  }

// Procedure: Get3DTINfile
void TTinGen3D :: Get3DTINfile()
  {
   char TENfilename [] = "c:\\data\\pnt24.ten";
   TENfile = fopen(TENfilename, "w");

   char* charN1 = "Node1";
   char* charN2 = "Node2";
   char* charN3 = "Node3";
   char* charN4 = "Node4";

   fprintf(TENfile, "%8s %8s %8s %8s\n", charN1, charN2, charN3, charN4);
   if (TENfile == NULL)
     {
      cerr << "Could not open 3D TIN file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The 3D TIN file is okay for writing ... " << endl;
     }
  }

// Function: Less
bool TTinGen3D :: Less(DataType a, DataType b)
  {
   return (a < b);
  }

// Function: Greater
bool TTinGen3D :: Greater(DataType a, DataType b)
  {
   return (a > b);
  }

//Procedure: Swap
void TTinGen3D :: Swap(DataType& a, DataType& b)
  {
   DataType temp;
```

```
    temp = a;
    a = b;
    b = temp;
   }

// Procedure: NodeOrder
void TTinGen3D :: NodeOrder(DataType& a, DataType& b, DataType& c, DataType&
d)
   {
    if (Greater(a, b))
        Swap(a, b);
    if (Greater(b, c))
        Swap(b, c);
    if (Greater(c, d))
        Swap(c, d);
    if (Greater(a, b))
        Swap(a, b);
   }

// Procedure: AddTENtoFile
void TTinGen3D :: AddTENtoFile(TVertex TEN)
   {
    fprintf(TENfile, "%8d %8d %8d %8d \n", TEN.N1, TEN.N2, TEN.N3, TEN.N4);
   }

//Procedure: GetSubImage
void TTinGen3D :: GetSubImage(int k, int i, int c, Voxel3D Voxel, Mask&
MaskPix)
   {
    int m, j;
    m = -1;
    for (k = 0; k < maxlevel-1; k ++)
      {
       for (i = 0; i < maxrow-1; i ++)
        {
         for (j = c; j < c + 1; j ++)
           {
            m ++;
            MaskPix[m] = Voxel[k][i][j];
           }
        }
      }
   }

// Procedure: GetMask
void TTinGen3D :: GetMaskVox(int l, int r, int c, Voxel3D Voxel, Mask&
MaskPix)
   {
    MaskPix[0] = Voxel[l][r][c];
    MaskPix[1] = Voxel[l][r][c+1];
    MaskPix[2] = Voxel[l][r+1][c];
    MaskPix[3] = Voxel[l][r+1][c+1];
    MaskPix[4] = Voxel[l+1][r][c];
    MaskPix[5] = Voxel[l+1][r][c+1];
    MaskPix[6] = Voxel[l+1][r+1][c];
    MaskPix[7] = Voxel[l+1][r+1][c+1];
   }

// Procedure: DEtectTEN
void TTinGen3D :: DetectTEN(Mask MaskPix)
   {
```

```
      bool DetectTEN1, DetectTEN2, DetectTEN3, DetectTEN4, DetectTEN5,
DetectTEN6;
     bool DetectA, DetectB, DetectC, DetectD, DetectE, DetectF, DetectG,
DetectH;
   bool DetectI, DetectJ, DetectK, DetectL;

   DetectTEN1 = (MaskPix[0] != MaskPix[2]) &&
                (MaskPix[2] != MaskPix[3]) &&
                (MaskPix[3] != MaskPix[4]) &&
                (MaskPix[0] != MaskPix[3]) &&
                (MaskPix[0] != MaskPix[4]) &&
                (MaskPix[2] != MaskPix[4]);

   DetectTEN2 = (MaskPix[0] != MaskPix[1]) &&
                (MaskPix[1] != MaskPix[3]) &&
                (MaskPix[3] != MaskPix[4]) &&
                (MaskPix[0] != MaskPix[3]) &&
                (MaskPix[0] != MaskPix[4]) &&
                (MaskPix[1] != MaskPix[4]);

   DetectTEN3 = (MaskPix[2] != MaskPix[3]) &&
                (MaskPix[3] != MaskPix[4]) &&
                (MaskPix[4] != MaskPix[6]) &&
                (MaskPix[2] != MaskPix[4]) &&
                (MaskPix[2] != MaskPix[6]) &&
                (MaskPix[3] != MaskPix[6]);

   DetectTEN4 = (MaskPix[1] != MaskPix[3]) &&
                (MaskPix[3] != MaskPix[4]) &&
                (MaskPix[4] != MaskPix[5]) &&
                (MaskPix[1] != MaskPix[4]) &&
                (MaskPix[1] != MaskPix[5]) &&
                (MaskPix[3] != MaskPix[5]);

   DetectTEN5 = (MaskPix[3] != MaskPix[4]) &&
                (MaskPix[4] != MaskPix[5]) &&
                (MaskPix[5] != MaskPix[7]) &&
                (MaskPix[3] != MaskPix[5]) &&
                (MaskPix[3] != MaskPix[7]) &&
                (MaskPix[4] != MaskPix[7]);

   DetectTEN6 = (MaskPix[3] != MaskPix[4]) &&
                (MaskPix[4] != MaskPix[6]) &&
                (MaskPix[6] != MaskPix[7]) &&
                (MaskPix[3] != MaskPix[6]) &&
                (MaskPix[3] != MaskPix[7]) &&
                (MaskPix[4] != MaskPix[7]);

      DetectA = (MaskPix[0] != MaskPix[5]);
      DetectB = (MaskPix[0] != MaskPix[6]);
      DetectC = (MaskPix[0] != MaskPix[7]);
      DetectD = (MaskPix[1] != MaskPix[2]);
      DetectE = (MaskPix[1] != MaskPix[6]);
      DetectF = (MaskPix[1] != MaskPix[7]);
      DetectG = (MaskPix[2] != MaskPix[5]);
      DetectH = (MaskPix[2] != MaskPix[7]);
      DetectI = (MaskPix[5] != MaskPix[6]);

      DetectJ = (! ((MaskPix[0] != MaskPix[4]) && (MaskPix[0] == MaskPix[3])
                                                  && (MaskPix[4] ==
MaskPix[7])));
```

```
      DetectK = (! ((MaskPix[2] != MaskPix[3]) && (MaskPix[2] == MaskPix[4])
                                                    && (MaskPix[3] ==
MaskPix[5])));

      DetectL = (! ((MaskPix[1] != MaskPix[3]) && (MaskPix[1] == MaskPix[4])
                                                    && (MaskPix[3] ==
MaskPix[6])));


   nTEN = 0;
   if (DetectTEN1 && DetectB && DetectC && DetectD
                  && DetectE && DetectG && DetectL)

       {
        Element.N1 = MaskPix[0];
        Element.N2 = MaskPix[2];
        Element.N3 = MaskPix[3];
        Element.N4 = MaskPix[4];
        nTEN ++;
        AddTENtoFile(Element);
        FoundTEN = true;
       }

   else
   if (DetectTEN2 && DetectA && DetectC && DetectD
                  && DetectE && DetectG && DetectK)

       {
        Element.N1 = MaskPix[0];
        Element.N2 = MaskPix[1];
        Element.N3 = MaskPix[3];
        Element.N4 = MaskPix[4];
        nTEN ++;
        AddTENtoFile(Element);
        FoundTEN = true;
       }
   else
   if (DetectTEN3 && DetectB && DetectC && DetectE
                  && DetectG && DetectH && DetectJ)

       {
        Element.N1 = MaskPix[2];
        Element.N2 = MaskPix[3];
        Element.N3 = MaskPix[4];
        Element.N4 = MaskPix[6];
        nTEN ++;
        AddTENtoFile(Element);
        FoundTEN = true;
       }
    else
    if (DetectTEN4 && DetectA && DetectC && DetectE
                   && DetectF && DetectG && DetectJ)

       {
        Element.N1 = MaskPix[1];
        Element.N2 = MaskPix[3];
        Element.N3 = MaskPix[4];
        Element.N4 = MaskPix[5];
        nTEN ++;
```

```
        AddTENtoFile(Element);
        FoundTEN = true;
       }
    else
    if (DetectTEN5 && DetectC && DetectE && DetectG
                   && DetectH && DetectI && DetectL)

       {
        Element.N1 = MaskPix[3];
        Element.N2 = MaskPix[4];
        Element.N3 = MaskPix[5];
        Element.N4 = MaskPix[7];
        nTEN ++;
        AddTENtoFile(Element);
        FoundTEN = true;
       }
    else
    if (DetectTEN6 && DetectC && DetectE && DetectG
                   && DetectH && DetectI && DetectK)

       {
        Element.N1 = MaskPix[3];
        Element.N2 = MaskPix[4];
        Element.N3 = MaskPix[6];
        Element.N4 = MaskPix[7];
        nTEN ++;
        AddTENtoFile(Element);
        FoundTEN = true;
       }
    else
    FoundTEN = false;
  }


// Procedure: ScanVoxels
void TTinGen3D :: ScanVoxels(Voxel3D Voxel)
  {
   int l, r, c;
   for (l = 1; l < maxlevel-1; l ++)
     {
      for (r = 1; r < maxrow-1; r ++)
        {
          for (c = 1; c < maxcol-1; c ++)
            {
             GetMaskVox(l, r, c, Voxel, MaskPix);
             DetectTEN(MaskPix);
            }
        }
     }
  }

// Procedure: Make3DTIN
void TTinGen3D :: Make3DTIN()
  {
   cout << "Reading the data ..., wait !" << endl;
   ReadVoxelImage(Voxel);
   cout << "Making 3D TIN structure ..." << endl;
   ScanVoxels(Voxel);
   cout << endl;
   cout << "Writing 3D TIN table to file ... " << endl;
   fclose(TENfile);
```

```
  }

//Procedure: Title
void TTinGen3D :: Title()
  {
   cout << endl;
    cout << "-------------Object-Oriented 3D Voronoi-to-3D TIN --------" <<
endl;
   cout << "----------- By: Alias Abdul-Rahman, July, 1998 -----------" <<
endl;
   cout << "           Input : .vpd, .vpi files. " << endl;
   cout << "          Output: 3D TIN file (.ten), an ASCII file. " << endl;
   cout << endl;
  }
```

```
// tinwin2d.rh

#define IDI_ICON1 1
#define CM_CLEAR            1125
#define CM_FILEEXIT         1126
#define CM_ABOUT            1127
#define CM_INPUTFILESITEM1     18874 // for XYZ file
#define CM_POPUPITEM1       18873 // for TIN file
```

```
//*********************************************************//
// A Windows program for TIN display                       //
//   Input: XYZ coordinates and TIN file                   //
//   Output: TINs in Windows environment                   //
//*********************************************************//

#include <owl/pch.h>
#include <owl/applicat.h>
#include <owl/framewin.h>
#include <owl/decframe.h>
#include <owl/statusba.h>
#include <owl/controlb.h>
#include <owl/buttonga.h>
#include <owl/gdiobjec.h>
#include <owl/chooseco.h>
#include <owl/inputdia.h>
#include <owl/opensave.h>
#include <owl/dc.h>
#include <mem.h>

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <string.h>

#include "tinwin2d.rh"

#define maxtriangle 20000
#define maxpoint 8000
#define maxarc 200

class TTINDrawApp : public TApplication
  {
   public:
      TTINDrawApp() : TApplication() {}  // call base class constructor
      void InitMainWindow();
  };

class TTINWindow : public TWindow
  {
   public:
      TTINWindow(TWindow* parent = 0)
         : TWindow(parent) {}

      struct Point
        {
         float x;
         float y;
         float z;
        };

      struct Triangle
        {
         int Node1;
         int Node2;
         int Node3;
        };

      Point* pnt_ptr[maxpoint];
```

```
      Triangle* tri_ptr[maxtriangle];

      struct Polygon
        {
         int Snode;
         int Enode;
        };

      Polygon* poly_ptr[maxarc];

      FILE* XYZfile;
      FILE* TINfile;
      FILE* ARCfile;

      TOpenSaveDialog :: TData fileData;
      int k, i, t, ii;
      float xmin, xmax, ymin, ymax;
      float zmin, zmax;
      float xlength, ylength;
      int npnt, ntri, narc;
      int MaxX, MaxY;
      float scaleX, scaleY;

      void ReadXYZ();
      void ReadTIN();
      void ReadARC();
      void GetMinMax(float&, float&,
                     float&, float&,
                     float&, float&);
      void Paint(TDC& dc, bool, TRect&);
      void EvSize(uint, TSize&);
      void CmInputFilesItem1();
      void CmPopupItem1();
      void CmFileExit()
           {PostQuitMessage(0);}
      void CmClear();
      void CmAbout();
      DECLARE_RESPONSE_TABLE(TTINWindow);

  };

DEFINE_RESPONSE_TABLE1(TTINWindow, TWindow)
  EV_WM_SIZE,
  EV_COMMAND(CM_INPUTFILESITEM1, CmInputFilesItem1),
  EV_COMMAND(CM_POPUPITEM1, CmPopupItem1),
  EV_COMMAND(CM_ABOUT, CmAbout),
  EV_COMMAND(CM_FILEEXIT, CmFileExit),
  EV_COMMAND(CM_CLEAR, CmClear),
END_RESPONSE_TABLE;


void TTINWindow :: CmInputFilesItem1()
  {
   ReadXYZ();
  }

void TTINWindow :: CmPopupItem1()
  {
   ReadTIN();
  }
```

```
void TTINWindow :: ReadXYZ()
  {
   string charX, charY, charZ;

   // Allocate memory for XYZ points
   for (i = 0; i < maxpoint; i ++)
     {
      pnt_ptr[i] = new Point;
     }

   // Reads again the the XYZ file
   //ifstream XYZfile = "c:\\data\\lochgrd.xyz";
   //ifstream XYZfile = "c:\\data\\lochass.xyz";
   //ifstream XYZfile = "c:\\data\\ard.xyz";
   ifstream XYZfile = "c:\\data\\pent.xyz";

   if (! XYZfile)
     {
      MessageBox ("Unable to open file: XYZ file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);
     }
   else
     {
      // reads the file header
      XYZfile >> charX >> charY >> charZ;
      npnt = 0;
        do
         {
          npnt ++;
          XYZfile >> pnt_ptr[npnt] -> x
                  >> pnt_ptr[npnt] -> y
                  >> pnt_ptr[npnt] -> z;

         } while (! XYZfile.eof());

     //if (XYZfile.eof() == true)
        //MessageBox ("Finished read the XYZ points", "XYZ file complete",
                  //MB_OK | MB_ICONINFORMATION);
     }
   XYZfile.close();
  }

void TTINWindow :: ReadTIN()
  {
   string charNode1, charNode2, charNode3;

   // Allocate memory for Triangle
   for (t = 0; t < maxtriangle; t ++)
     {
      tri_ptr[t] = new Triangle;
     }

   // Reads the TIN file
   //ifstream TINfile = "c:\\data\\lochgrd.tin";
   //ifstream TINfile = "c:\\data\\lochas2.tin";
   //ifstream TINfile = "c:\\data\\ard.tin";
   ifstream TINfile = "c:\\data\\pent.tin";


   if (!TINfile)
     {
```

```
      MessageBox("Unable to open file: TIN file",
                 "File Error", MB_OK | MB_ICONEXCLAMATION);
    }

  else
    {
     ntri = 0;

     // reads the file header
     TINfile >> charNode1 >> charNode2 >> charNode3;
     do
       {
        ntri ++;
        TINfile >> tri_ptr[ntri] -> Node1
                >> tri_ptr[ntri] -> Node2
                >> tri_ptr[ntri] -> Node3;

       } while (! TINfile.eof());

      //if (TINfile.eof() == true)
       //MessageBox ("Finished read the TIN nodes", "TIN file complete",
                  //MB_OK | MB_ICONINFORMATION);
     }

  TINfile.close();
 }

void TTINWindow :: ReadARC()
 {
  string charSnode, charEnode;

  // Allocate memory for XYZ points
  for (int ii = 0; ii < maxarc; ii ++)
    {
     poly_ptr[ii] = new Polygon;
    }

  // Reads again the the XYZ file
  ifstream ARCfile = "c:\\data\\lake.arc";


  if (! ARCfile)
    {
     MessageBox ("Unable to open file: ARC file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);
    }
  else
    {
     // reads the file header
     ARCfile >> charSnode >> charEnode;
     narc = 0;
       do
        {
         narc ++;
         ARCfile >> poly_ptr[narc] -> Snode
                 >> poly_ptr[narc] -> Enode;

        } while (! ARCfile.eof());

    //if (ARCfile.eof() == true)
       //MessageBox ("Finished read the ARC points", "ARC file complete",
```

```
                    //MB_OK | MB_ICONINFORMATION);
      }
    ARCfile.close();
  }


void TTINWindow :: CmClear()
  {
   // to clear and redraw
   Invalidate();
  }


void TTINWindow :: CmAbout()
  {
   MessageBox ("TINs display with menu - by Alias Abdul-Rahman, 1999",
               "About the Program", MB_OK | MB_ICONINFORMATION);
  }

void TTINWindow :: GetMinMax(float& xmax, float& ymax,
                             float& xmin, float& ymin,
                             float& xlength, float& ylength)
  {
   string charX, charY, charZ;

   // Read the XYZ file and get the min-max of the coordinates
   //ifstream XYZfile = "c:\\data\\lochgrd.xyz";
   //ifstream XYZfile = "c:\\data\\lochass.xyz";
   //ifstream XYZfile = "c:\\data\\ard.xyz";
   ifstream XYZfile = "c:\\data\\pent.xyz";

   if (! XYZfile)
      MessageBox ("Unable to open file: XYZ file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);

   float x, y, z;

   // reads the file header
   XYZfile >> charX >> charY >> charZ;

   XYZfile >> x >> y >> z;
      xmin = x;
      xmax = x;

      ymin = y;
      ymax = y;

      zmin = z;
      zmax = z;

      do
        {
         XYZfile >> x >> y >> z;

         if (xmin > x)
             xmin = x;

         if (ymin > y)
             ymin = y;

         if (zmin > z)
```

```
            zmin = z;

        if (xmax < x)
            xmax = x;

        if (ymax < y)
            ymax = y;

        if (zmax < z)
            zmax = z;
        } while (! XYZfile.eof());

      xlength = xmax - xmin;
      ylength = ymax - ymin;
  }

void TTINWindow :: Paint(TDC& dc, bool, TRect&)
  {
   GetMinMax(xmax, ymax, xmin, ymin, xlength, ylength);

   ReadXYZ(); // reads the XYZ
   ReadTIN(); // reads the TIN
   //ReadARC(); // reads the ARC

   // to set the background color of the client area
   // comment out the next line for default colour
   //static const TColor color (RGB(0, 0, 0));      // RGB (0, 0, 0) - black
   //TWindow :: SetBkgndColor(color);               // RGB (192, 192, 192) -
gray


   RECT rect;
   :: GetClientRect(HWindow, &rect);

   dc.SetMapMode(MM_ANISOTROPIC);

   scaleX = (rect.right - rect.left) / xlength;
   scaleY = (rect.bottom - rect.top) / ylength;

   for (k = 1; k < npnt; k ++)
     {
      pnt_ptr[k] -> x = (pnt_ptr[k] -> x - xmin) * scaleX;
      pnt_ptr[k] -> y = (rect.bottom - rect.top) - ((pnt_ptr[k] -> y - ymin)
                                                    * scaleY);
     }

   // set pen for points display
   TPen pen1 = TPen(RGB(255, 0, 0), 1, PS_SOLID);
   SelectObject(dc, pen1);
   for (k = 1; k < npnt; k ++)
     {
      dc.MoveTo(pnt_ptr[k] -> x, pnt_ptr[k] -> y);
      dc.LineTo(pnt_ptr[k] -> x, pnt_ptr[k] -> y);
     }

   SelectObject(dc, pen1);
   DeleteObject(pen1);


   // set pen style for TINs display
   TPen pen2 = TPen(RGB(0, 0, 255), 1, PS_SOLID);
```

```
   SelectObject(dc, pen2);

   for (k = 1; k < ntri; k ++)
     {
      dc.MoveTo(pnt_ptr[tri_ptr[k] -> Node1] -> x,
                pnt_ptr[tri_ptr[k] -> Node1] -> y);

      dc.LineTo(pnt_ptr[tri_ptr[k] -> Node2] -> x,
                pnt_ptr[tri_ptr[k] -> Node2] -> y);
      dc.LineTo(pnt_ptr[tri_ptr[k] -> Node3] -> x,
                pnt_ptr[tri_ptr[k] -> Node3] -> y);
      dc.LineTo(pnt_ptr[tri_ptr[k] -> Node1] -> x,
                pnt_ptr[tri_ptr[k] -> Node1] -> y);
     }
   SelectObject(dc, pen2);
   DeleteObject(pen2);



   // set pen style for polygon (i.e. constrained edges) display
   TPen pen3 = TPen(RGB(255, 0, 0), 2, PS_SOLID);
   SelectObject(dc, pen3);

   for (int r = 1; r < narc; r ++)
     {
      dc.MoveTo(pnt_ptr[poly_ptr[r] -> Snode] -> x,
                pnt_ptr[poly_ptr[r] -> Snode] -> y);
      dc.LineTo(pnt_ptr[poly_ptr[r] -> Enode] -> x,
                pnt_ptr[poly_ptr[r] -> Enode] -> y);
      dc.LineTo(pnt_ptr[poly_ptr[r] -> Snode] -> x,
                pnt_ptr[poly_ptr[r] -> Snode] -> y);
     }
   SelectObject(dc, pen3);
   DeleteObject(pen3);



   // Deallocate memory
   for (i = 0; i < maxpoint; i ++)
     {
      delete [] pnt_ptr[i];
     }

   for (t = 0; t < maxtriangle; t ++)
     {
      delete [] tri_ptr[t];
     }

   for (ii = 0; ii < maxarc; ii ++)
     {
      delete [] poly_ptr[ii];
     }

  }

// Force the window to repaint if resize
void TTINWindow :: EvSize(uint, TSize&)
  {
   :: InvalidateRect(HWindow, 0, true);
  }
```

```
void TTINDrawApp :: InitMainWindow()
  {
   // construct the decorated MDI frame window
   TDecoratedFrame* frame = new TDecoratedFrame(0,
                      "TINSoft ver. 1.0", new TTINWindow);

   // construct the status bar
   TStatusBar* sb = new TStatusBar(frame, TGadget :: Recessed,
                                    TStatusBar :: CapsLock | TStatusBar ::
NumLock);

   // construct the control bar
   TControlBar* cb = new TControlBar(frame);

   cb -> Insert(*new TButtonGadget(CM_INPUTFILESITEM1, CM_INPUTFILESITEM1,
                      TButtonGadget :: Command));
   cb -> Insert(*new TButtonGadget(CM_POPUPITEM1, CM_POPUPITEM1,
                      TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_CLEAR, CM_CLEAR,
                      TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_FILEEXIT, CM_FILEEXIT,
                      TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_ABOUT, CM_ABOUT,
                      TButtonGadget :: Command));

   cb -> SetHintMode(TGadgetWindow :: EnterHints);

   // set client area to the application workspace
   frame -> SetBkgndColor(::GetSysColor(COLOR_APPWORKSPACE));

   // insert the status and control bar into the frame
   frame -> Insert(*sb, TDecoratedFrame :: Bottom);
   frame -> Insert(*cb, TDecoratedFrame :: Top);

   // set the main window and its menu
   SetMainWindow(frame);
   GetMainWindow() -> AssignMenu("COMMANDS");
   EnableCtl3d(true);
  }


int OwlMain(int /*argc*/, char* /*argv*/ [])
  {
   return TTINDrawApp().Run();
  }
```

```
#include "tinwin2d.rh"

COMMANDS MENU
{
 POPUP "&InputFiles"
 {
  MENUITEM "&XYZ File", CM_INPUTFILESITEM1
  MENUITEM "&TIN File", CM_POPUPITEM1
 }

 POPUP "&Redraw"
 {
  MENUITEM "&Clear and Redraw", CM_CLEAR
  MENUITEM "E&xit", CM_FILEEXIT
 }

 MENUITEM "&About", CM_ABOUT
}

STRINGTABLE
  {
   CM_INPUTFILESITEM1,  "Get the XYZ file"
   CM_POPUPITEM1,       "Get the TIN file"
   CM_CLEAR,            "Redraw the TINs"
   CM_FILEEXIT,         "Exit the program ..."
   CM_ABOUT,            "About the program ..."
  }

CM_CLEAR BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 91 77 77 77 77 77 77 77 77 99'
 '00 00 79 77 77 77 77 77 77 77 79 17 00 00 77 17'
 '77 87 77 77 77 79 91 77 00 00 77 77 80 08 88 87'
 '77 77 77 77 00 00 77 78 00 00 08 88 77 77 77 77'
 '00 00 77 78 0A 0A A0 08 88 87 77 77 00 00 77 78'
 '0A 0A AA A0 88 88 77 77 00 00 77 80 AA 0A AA AA'
 '00 88 77 77 00 00 77 80 AA A0 AA A0 A0 88 77 77'
 '00 00 77 0A AA A0 AA 0A A0 88 71 17 00 00 77 0A'
 'AA A0 00 AA A0 88 79 97 00 00 78 0A AA A0 AA AA'
 'A0 88 77 77 00 00 78 00 00 00 AA AA A0 88 77 77'
 '00 00 78 88 87 78 0A AA A0 88 77 77 00 00 77 77'
 '77 77 80 AA A0 88 77 77 00 00 77 71 77 77 78 0A'
 'A0 88 77 77 00 00 77 19 77 77 77 80 A0 88 77 77'
 '00 00 77 97 77 77 77 78 00 88 79 77 00 00 71 77'
 '77 77 77 77 88 87 71 19 00 00 19 77 77 77 77 77'
 '77 77 77 71 00 00'
}


CM_FILEEXIT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
```

```
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 EE EE EE EE EE EE EE EE EE EE'
 '00 00 EE EE FF EF EF FE EE EE EE EE 00 00 EE EE'
 'FF EE FE EF EE EE EE EE 00 00 EE EE FF EF FF FF'
 'EE EE EE EE 00 00 EE EE EF EE EF FF EE EE EE EE'
 '00 00 88 88 88 88 88 88 88 88 88 88 00 00 66 66'
 '8A AA A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A A9 A8 77 86 66 66 66 00 00 66 66'
 '8A A9 A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A AA A8 77 86 66 66 66 00 00 66 66'
 '8A A8 87 77 86 66 66 66 00 00 66 66 8A 87 77 77'
 '86 66 66 66 00 00 66 66 88 77 77 77 86 66 66 66'
 '00 00 66 66 88 88 88 88 86 66 66 66 00 00 66 66'
 '66 66 66 66 66 66 66 66 00 00 66 66 66 66 66 66'
 '66 66 66 66 00 00'
}


CM_ABOUT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB B8 88 88 BB BB BB BB'
 '00 00 BB BB BB B8 00 88 BB BB BB BB 00 00 BB BB'
 'BB B8 00 BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00 BB BB BB B8 00 88 BB BB BB BB'
 '00 00 BB BB BB BB 80 08 BB BB BB BB 00 00 BB BB'
 'B8 8B B8 00 8B BB BB BB 00 00 BB BB B8 00 BB 00'
 '8B BB BB BB 00 00 BB BB B8 00 BB 00 8B BB BB BB'
 '00 00 BB BB BB 80 00 00 8B BB BB BB 00 00 BB BB'
 'BB B8 00 00 8B BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00'
}

CM_INPUTFILESITEM1 BITMAP "pointfile.bmp"
/*{
 '42 4D 96 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 18 00 00 00 18 00 00 00 01 00 04 00 00 00'
 '00 00 20 01 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF FF FF FF FF FF FF FF FF'
```

```
 'FF FF FF 00 00 00 00 00 00 00 00 00 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 0A AA AA AA AA'
 'AA AA AA AA 08 8F FF 0A 99 9A AA AA AA AA AA AA'
 '08 8F FF 0A AA AA AA AA AA AA AA AA 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 00 00 00 00 00'
 '00 00 00 00 08 8F FF 88 88 88 88 88 88 88 88 88'
 '88 8F FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF F0 00 00 00 00'
 '00 00 0F FF FF FF FF F0 FF FF FF FF FF FF 0F FF'
 'FF FF FF F0 F0 00 FF FF FF FF 0F FF FF FF FF F0'
 'FF FF FF F0 00 0F 0F FF FF FF FF F0 FF FF FF FF'
 'FF FF 0F FF FF FF FF F0 FF FF FF 00 0F FF 0F FF'
 'FF FF FF F0 FF FF FF FF FF F0 8F FF FF FF FF F0'
 'F0 00 FF FF FF 08 8F FF FF FF FF F0 FF FF FF FF'
 '00 88 8F FF FF FF FF F0 FF FF FF F0 88 88 8F FF'
 'FF FF FF F0 00 00 00 08 88 88 8F FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF'
}
*/

CM_POPUPITEM1 BITMAP "tinfile.bmp"
/*{
 '42 4D 96 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 18 00 00 00 18 00 00 00 01 00 04 00 00 00'
 '00 00 20 01 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF 00 00 00 00 00 00 00 00 00 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 0A AA AA AA AA'
 'AA AA AA AA 08 8F FF 0A A9 99 9A AA AA AA AA AA'
 '08 8F FF 0A AA AA AA AA AA AA AA AA 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 00 00 00 00 00'
 '00 00 00 00 08 8F FF 88 88 88 88 88 88 88 88 88'
 '88 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00'
 '00 00 00 FF FF FF FF FF 0F FF FF FF FF FF F0 FF'
 'FF FF FF FF 0F FF FF FF 00 FF F0 FF FF FF FF FF'
 '0F FF FF FF FF FF F0 FF FF FF FF FF 0F F0 00 0F'
 'FF FF F0 FF FF FF FF FF 0F FF FF FF FF FF F0 FF'
 'FF FF FF FF 0F FF FF F0 0F FF F0 FF FF FF FF FF'
 '0F FF FF FF FF FF 08 FF FF FF FF FF 0F F0 00 FF'
 'FF F0 88 FF FF FF FF FF 0F FF FF FF F0 08 88 FF'
 'FF FF FF FF 0F FF FF FF 08 88 88 FF FF FF FF FF'
 '00 00 00 00 88 88 88 FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF'
}

*/
```

TIN Data Restructuring Code

The following gives the complete codes for the data restructuring, that is from the normal TIN structure to triangle's sides structure. This is for the purpose of TIN-based applications such as contouring.

```
//*************************************************************//
//       A program to restruct TIN structure to SID structure    //
//         Input : TIN file, NBR file                            //
//         Output : Sides file                                   //
//                                                               //
//*************************************************************//

#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string>

#define maxtriangle 15000
#define maxside 20000


class Restruct
  {
   public:
      // Triangle table structure
      struct ttri
        {
         int Node1;
         int Node2;
         int Node3;
        };
      ttri* Tri3Nodes[maxtriangle];

      // Triangle neighbour structure
      struct tnbr
        {
         int trino;
         int numnbr;
         int Nbr1, Nbr2, Nbr3;
        };
      tnbr* TriNbr[maxtriangle];

      // Triangle sides structure
      struct tsid
        {
         int Node1;
         int Node2;
```

```
        int RightTri;
        int LeftTri;
       };
     tsid* TriSides[maxside];

    // structure for 3 sides table
    struct trs
      {
       int Side[3];
      };
    trs* Tri3Sides[maxtriangle];

    int ntri;
    int nsid;
    int idx[maxside];
    int ridx[maxside];

    FILE* TINfile;
    FILE* NBRfile;

    Restruct();    // constructor
    ~Restruct();   // destructor

    void PressAnyKey();
    void ReadTINFile();
    void ReadNBRFile();
    void WriteOutputFile();
    bool ExistingSides(int, int, int&);
    void DoSide(int, int, int, int);
    void MakeSides();
    bool Less(int, int);
    void Swap(int, int);
    void QuickSort(int, int);
    void Sort();
  };


// some definitions
Restruct :: Restruct()          // constructor
  {
   // Allocate memory for TIN data
   for (int t = 0; t < maxtriangle; t ++)
     {
      Tri3Nodes[t] = new ttri;
     }

   // Allocate memory for TIN neighbour data
   for (int n = 0; n < maxtriangle; n ++)
     {
      TriNbr[n] = new tnbr;
     }

   // Allocate memory for Triangle sides data
   for (int s = 0; s < maxtriangle; s ++)
     {
      TriSides[s] = new tsid;
     }

   for (int k = 0; k < maxtriangle; k ++)
     {
```

```
      Tri3Sides[k] = new trs;
     }
  }

Restruct :: ~Restruct()        // destructor
  {
   // Deallocate memory for TIN data
   for (int r = 0; r < maxtriangle; r ++)
     {
      delete [] Tri3Nodes[r];
     }

   // Allocate memory for TIN neighbour data
   for (int n = 0; n < maxtriangle; n ++)
     {
      delete [] TriNbr[n];
     }

   // Allocate memory for Triangle sides data
   for (int s = 0; s < maxtriangle; s ++)
     {
      delete [] TriSides[s];
     }

   for (int k = 0; k < maxtriangle; k ++)
     {
      delete [] Tri3Sides[k];
     }
  }


void Restruct :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

void Restruct :: ReadTINFile()
  {
   string charNode1, charNode2, charNode3;
   ifstream TINfile("c:\\data\\sercomt$.tin");

   if (! TINfile)
     cout << "Unable to open TIN file" << endl;
   else
     {
      cout << "TIN file is opened and okay for reading." << endl;
      cout << endl;

      // reads the file header
      TINfile >> charNode1 >> charNode2 >> charNode3;
      ntri = 0;
      do
        {
         //ntri ++;
         TINfile >> Tri3Nodes[ntri]->Node1
                 >> Tri3Nodes[ntri]->Node2
                 >> Tri3Nodes[ntri]->Node3;
         ntri ++;
```

```
        } while (! TINfile.eof());
      }

    cout << endl;
    cout << "ntri: " << ntri << endl;
    TINfile.close();
   }

void Restruct :: ReadNBRFile()
   {
    string chartrino;
    string charnumnbr;
    string charNbr1;
    string charNbr2;
    string charNbr3;

    ifstream NBRfile("c:\\data\\sercomt$.nbr");

    if (! NBRfile)
      cout << "Unable to open file" << endl;
    else
      {
       cout << "TIN NBR file is opened and okay for reading." << endl;
       cout << endl;

       NBRfile >> chartrino >> charnumnbr
               >> charNbr1 >> charNbr2 >> charNbr3;

       ntri = 0;
       do
         {
          ntri ++;
          NBRfile >> TriNbr[ntri]->trino
                  >> TriNbr[ntri]->numnbr
                  >> TriNbr[ntri]->Nbr1
                  >> TriNbr[ntri]->Nbr2
                  >> TriNbr[ntri]->Nbr3;

         } while (! NBRfile.eof());
      }

    NBRfile.close();
   }

void Restruct :: WriteOutputFile()
   {
    char* chartrino = "Tri#";
    char* charSide1 = "Side1";
    char* charSide2 = "Side2";
    char* charSide3 = "Side3";

    char* charsideno = "Side#";
    char* charNode1 = "Node1";
    char* charNode2 = "Node2";
    char* charRightTri = "RightTri";
    char* charLeftTri = "LeftTri";

    // write the tri sides output file
    char TRISIDESfilename [] = "c:\\data\\sercomt$.sid";
    FILE* TRISIDESfile = fopen(TRISIDESfilename, "w");
```

```
     fprintf(TRISIDESfile, "%5s %5s %5s %5s \n", chartrino, charSide1,
charSide2, charSide3);

   for (int i = 1; i <= ntri-1; i ++)
     {
      fprintf(TRISIDESfile, "%5d %5d %5d %5d \n", i, Tri3Sides[i]->Side[0],
                                                     Tri3Sides[i]->Side[1],
                                                    Tri3Sides[i]->Side[2]);
     }
   fclose(TRISIDESfile);

   // write the sides output file
   char SIDESfilename [] = "c:\\data\\sercomt$.trs";
   FILE* SIDESfile = fopen(SIDESfilename, "w");

     fprintf(SIDESfile, "%5s %5s %5s %5s %5s \n", charsideno, charNode1,
charNode2,
                                                  charRightTri, charLeftTri);

   for (int i = 1; i <= nsid; i ++)
     {
      fprintf(SIDESfile, "%5d %5d %5d %7d %8d \n", i, TriSides[i]->Node1,
                                                      TriSides[i]->Node2,
                                                     TriSides[i]->RightTri,
                                                     TriSides[i]->LeftTri);
     }
   fclose(SIDESfile);

  }


bool Restruct :: ExistingSides(int n1, int n2, int& s)
  {
   bool found;

   s = 0;
   found = false;

   do
     {
      found = (n1 == TriSides[s]->Node1) && (n2 == TriSides[s]->Node2);
      if (! found)
        s ++;
     } while ((! found) && (s <= nsid));

   return found;
  }

void Restruct :: DoSide(int t, int n1, int n2, int snbr)
  {
   int h;
   int s;

   if (n1 > n2)
     {
      h = n1;
      n1 = n2;
      n2 = h;
     }

   if (ExistingSides(n1, n2, s))
```

```
      {
       TriSides[s]->RightTri = t;
       Tri3Sides[t]->Side[snbr] = s;
      }

    else
      {
       nsid ++;
       TriSides[nsid] = new tsid;
       TriSides[nsid]->Node1 = n1;
       TriSides[nsid]->Node2 = n2;
       TriSides[nsid]->LeftTri = t;
       TriSides[nsid]->RightTri = 0;

       Tri3Sides[t]->Side[snbr] = nsid;
      }
  }


void Restruct :: MakeSides()
  {
   ifstream TINfile("c:\\data\\sercomt$.tin");
   nsid = 0;

   for (int t = 1; t < ntri; t ++)
     {
              TINfile  >>  Tri3Nodes[t]->Node1  >>  Tri3Nodes[t]->Node2  >>
Tri3Nodes[t]->Node3;
      printf("\r");
      cout << "t: " << t;
      DoSide(t, (int)Tri3Nodes[t]->Node1, (int)Tri3Nodes[t]->Node2, 0);
      DoSide(t, (int)Tri3Nodes[t]->Node2, (int)Tri3Nodes[t]->Node3, 1);
      DoSide(t, (int)Tri3Nodes[t]->Node3, (int)Tri3Nodes[t]->Node1, 2);
     }

   TINfile.close();
  }

bool Restruct :: Less(int i, int j)
  {
   int ni1;
   int ni2;
   int nj1;
   int nj2;

   ni1 = (int)TriSides[idx[i]]->Node1;
   ni2 = (int)TriSides[idx[i]]->Node2;
   nj1 = (int)TriSides[idx[j]]->Node1;
   nj2 = (int)TriSides[idx[j]]->Node2;

   return (ni1 < nj1) || ((ni1 == nj1) && (ni2 < nj2));
  }

void Restruct :: Swap(int i, int j)
  {
   int h;
   h = idx[i];
   idx[j] = h;
  }

void Restruct :: QuickSort(int l, int r)
```

```
  {
   int m;
   int i;
   int j;

   i = l;
   j = r;
   m = (l + r) / 2;

   do
     {
      do
        {
         i ++;
        } while (Less(i, m));

      do
        {
         j --;
        } while (Less(m, j));

      if (i <= j)
        {
         if (i == m)
           m = j;
         else
           if (j == m)
             m = i;
         Swap(i, j);
         i ++;
         j ++;
        }

     } while (i > j);

   if (l < j)
      QuickSort(l, j);
   if (i < r)
      QuickSort(i, r);
  }

void Restruct :: Sort()
  {
   int i;
   int j;

   for (i = 0; i < nsid; i ++)
     {
      idx[i] = i;
     }

   QuickSort(1, nsid);
   for (i = 0; i < nsid; i ++)
     {
      ridx[idx[i]] = i;
     }

   for (i = 0; i < ntri; i ++)
     {
      for (j = 0; j <= 2; j ++)
```

```
            Tri3Sides[i]->Side[j] = ridx[Tri3Sides[i]->Side[j]];
        }
    }

// main program
void main()
    {
     Restruct Re;
     Re.ReadTINFile();
     Re.ReadNBRFile();
     cout << "Processing ...." << endl;
     Re.MakeSides();
     cout << endl;
     //cout << "Sort ..." << endl;
     //Re.Sort();
     cout << endl;
     cout << "Output ..." << endl;
     Re.WriteOutputFile();
    }
```

TIN Topology Code

The following gives the complete codes for the TIN neighbouring program.

```
//************************************************//
//       A program to generate TIN topology       //
//         Input : TIN file                       //
//         Output : TIN neighbours (.NBR) file    //
//                                                //
//************************************************//

#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string>

#define maxtriangle 15000

int evencheck(int value)
  {
   return (value & 1) == 0;
  }

class Neighbour
  {
   public:
     int TriNum;
     int TotalNeighbour;
     int Nbr[3];
     Neighbour()
       {
        TriNum = 0;
        TotalNeighbour = 0;
        Nbr[0] = Nbr[1] = Nbr[2] = 0;
       };
     ~Neighbour(){};
  };


class TinNeighbour
  {
   public:
      // structure for the TINs
      struct ThreeNodes
        {
         int Node[3];
        };
      ThreeNodes* tri[maxtriangle];

     // some global variables
```

```
    FILE* TINfile;
    FILE* NBRfile;
    int t;
    int ntri;
    bool isTriangleNode;

    // some operations
    TinNeighbour();
    ~TinNeighbour();

    void GetTINfile();
    void GetNeighbourfile();
    bool Less(int a, int b);
    bool Greater(int, int);
    void Swap(int&, int&);
    void NodeOrder(int&, int&, int&);
    void AddNbrtoFile(Neighbour);
    void MakeTinNeighbour();
    void PressAnyKey();
};

TinNeighbour :: TinNeighbour()
  {
   // Allocate memory for TINs
   for (int t = 0; t < maxtriangle; t ++)
     {
      tri[t] = new ThreeNodes;
     }
  }

TinNeighbour :: ~TinNeighbour()
  {
   // Deallocate memory for TIN neighbour
   for (int t = maxtriangle-1; t >= 0; t --)
     {
      delete [] tri[t];
     }
  }

void TinNeighbour :: GetTINfile()
  {
   int t;
   string charNode1, charNode2, charNode3;

   // Reads the TIN file
   //ifstream TINfile = "c:\\data\\sercomt$.tin";
   ifstream TINfile = "c:\\data\\lakebo.tin";

   if (TINfile == NULL)
     {
      cerr << "Could not open TIN file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The TIN file is opened" << endl;

      // reads the file header
      TINfile >> charNode1 >> charNode2 >> charNode3;
```

```
      t = 0;
      do
        {
         printf("\r");
         t++;
         cout << "reads Triangle  ... " << t;

         for (int n = 0; n < 3; n ++)
           {
            TINfile >> tri[t] -> Node[n];
           }

        } while (! TINfile.eof());
     }
    ntri = t-1;
    TINfile.close();
  }

void TinNeighbour :: GetNeighbourfile()
  {
   //char Neighbourfilename [] = "c:\\data\\sercomt$.nbr";
   //char Neighbourfilename [] = "c:\\data\\lakebo.nbr";

   NBRfile = fopen(Neighbourfilename, "w");

   char* TINnum = "Tri#";
   char* NumofNbr = "NumofNeighbours";
   char* Nbr1 = "Nbr1";
   char* Nbr2 = "Nbr2";
   char* Nbr3 = "Nbr3";

   fprintf(NBRfile, "%8s %8s %8s %8s %8s \n", TINnum, NumofNbr, Nbr1, Nbr2,
Nbr3);
   if (NBRfile == NULL)
     {
      cerr << "Could not open TIN Neighbour file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "preparing TIN Neighbour file ... " << endl;
      cout << endl;
     }
  }

// Function: Less
bool TinNeighbour :: Less(int a, int b)
  {
   return (a < b);
  }

// Function: Greater
bool TinNeighbour :: Greater(int a, int b)
  {
   return (a > b);
  }

//Procedure: Swap
void TinNeighbour :: Swap(int& a, int& b)
  {
```

```
    int temp;

    temp = a;
    a = b;
    b = temp;
   }

// Procedure: NodeOrder
void TinNeighbour :: NodeOrder(int& a, int& b, int& c)
   {
    if (Greater(a, b))
        Swap(a, b);
    if (Greater(b, c))
        Swap(b, c);
    if (Greater(a, b))
        Swap(a, b);
   }

void TinNeighbour :: AddNbrtoFile(Neighbour Element)
   {
    fprintf(NBRfile, "%7d %8d %14d %8d %8d \n", Element.TriNum,
                                                Element.TotalNeighbour,
                                                Element.Nbr[0],
                                                Element.Nbr[1],
                                                Element.Nbr[2]);
   }

void TinNeighbour :: MakeTinNeighbour()
   {
    int CommonNode;
    int NumofNbr;
    bool isTriangleNode;

    int tt;
    int i;

    for (t = 1; t <= ntri; t ++)
       {
        printf("\r");
        cout << "Making TIN neighbours ... " << t;

        Neighbour Element;
        NumofNbr = -1;

        for (tt = 1; (tt <= ntri) && (NumofNbr <= 3); tt ++)
          {
           if (t == tt) continue;

           CommonNode = 0;

           for (i = 0; (i < 3) && (CommonNode <= 2); i ++)
               {
                isTriangleNode = (tri[t] -> Node[i] == tri[tt] -> Node[0]) ||
                                 (tri[t] -> Node[i] == tri[tt] -> Node[1]) ||
                                 (tri[t] -> Node[i] == tri[tt] -> Node[2]);

                if (isTriangleNode == true)
                  {
                   CommonNode ++;
                   if (CommonNode == 2)
                     {
```

```
                    NumofNbr ++;
                    Element.Nbr[NumofNbr] = tt;
                    Element.TotalNeighbour = NumofNbr + 1;
                   }
                }
              }
            }

        Element.TriNum = t;
        AddNbrtoFile(Element);
       }

   fclose(NBRfile);
  }

void TinNeighbour :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

// Main program
void main()
  {
   TinNeighbour TINNBR;
   TINNBR.GetTINfile();
   TINNBR.GetNeighbourfile();
   TINNBR.MakeTinNeighbour();
   TINNBR.PressAnyKey();
  }
```

Contouring Code

The following gives the complete code for the contouring from TIN data structures.

```
//*************************************************************//
//                                                             //
//       A Program to interpolate contours from TIN            //
//           Input: point (XYZ) file, TRS and SID files        //
//           Output: Contour interpolation file (.CON)         //
//                                                             //
//*************************************************************//

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <conio.h>
#include <math.h>
#include <string>

#define maxpoint 7000
#define maxtriangle 10000
#define maxside 16000
#define accy 0.001

class Contour
  {
   public:
      struct PointStruct
        {
         float x;
         float y;
         float z;
        };
      PointStruct* Point[maxpoint];

      struct TriSideStruct
        {
         int Node1;
         int Node2;
         int RightTri;
         int LeftTri;
        };
      TriSideStruct* TriSides[maxside];

      struct Tri3SidesStruct
        {
         int Side[3];
        };
      Tri3SidesStruct* Tri3Sides[maxtriangle];

      bool utri[maxtriangle];
```

```
      string charX;
      string charY;
      string charZ;
      int npnt, ntri, nside;
      int Side1, Side2, Side3;
      float x, y, z, xNd1, yNd1, xNd2, yNd2;
      float xmin, xmax, ymin, ymax;
      int interval;
      int SegNr, Hreq, prevH;
      float Hmin, Hmax, rangeH;
      float zNd1, zNd2;
      int ii;
      int t, s, i, j, k;
      FILE* interpolfile;

      void PressAnyKey();
      void PrintInput();
      void ReadInputFiles();
      void OpenOutputFile();
      void AddToFile(float, float);
      void GetMaxMinHeights();
      void GetCoordinate(int, float&, float&,
                         float&, float&,
                         float&, float&);
      void Interpolate(int, int, int);
      bool CheckSide(int);
      int FindFirstTri(int&, int&);
      void FindOtherSide(int, int, int&);
      void FindNextTri(int, int, int&);
      void GetContours(int);
      void GetInterval(int&);
      void MakeContouring();

      Contour();    // constructor
      ~Contour();   // deconstructor
  };

// Implementation of the Contour class
Contour :: Contour()
  {
   // Allocate memory for Points
   for (int i = 0; i < maxpoint; i ++)
       {
        Point[i] = new PointStruct;
       }

   // Allocate memory for Triangle sides
   for (int j = 0; j < maxside; j ++)
       {
        TriSides[j] = new TriSideStruct;
       }

   // Allocate memory for Trinagle 3 sides
   for (int k = 0; k < maxtriangle; k ++)
       {
        Tri3Sides[k] = new Tri3SidesStruct;
       }
  }

Contour :: ~Contour()
```

```
  {
   // Deallocate memory for points
   for (int i = 0; i < maxpoint; i ++)
       {
        delete [] Point[i];
       }

  // Deallocate memory for Triangles sides
  for (int j = 0; j < maxside; j ++)
       {
        delete [] TriSides[j];
       }

 // Deallocate memory for Triangle 3 sides
 for (int k = 0; k < maxtriangle; k ++)
       {
        delete [] Tri3Sides[k];
       }
  }


void Contour :: PressAnyKey()
  {
   cout << endl;
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

void Contour :: ReadInputFiles()  // Point, Sides, and Triangle sides files
  {
   // Reads the Point file
   ifstream pointfile("c:\\data\\sercomb.xyz");
     if (! pointfile)
       {
        cout << "Could not open point file !" << endl;
        exit(1);
       }
       else
         {
          cout << endl;
          cout << "The point file is opened" << endl;

          pointfile >> charX >> charY >> charZ;

          npnt = 0;
          do
            {
             npnt ++;
             printf("\r");
             cout << "Reading the points ... " << npnt;
            pointfile >> Point[npnt]->x >> Point[npnt]->y >> Point[npnt]->z;
             //npnt ++;
            } while (! pointfile.eof());
          npnt = npnt - 1;
         }

   // to print the node coordinates
   //cout << endl;
   //cout << "The nodes coordinates:" << endl;
```

```
//cout << Point[1]->x << " " << Point[1]->y <<" " << Point[1]->z << endl;
//cout << Point[2]->x << " " << Point[2]->y <<" " << Point[2]->z << endl;
//cout << Point[3]->x << " " << Point[3]->y <<" " << Point[3]->z << endl;
//cout << endl;
//PressAnyKey();

// Reads the Triangle sides file
string charSideNo;
string charNode1;
string charNode2;
string charRightTri;
string charLeftTri;

ifstream trisidfile("c:\\data\\sercomt$.trs");

   if (! trisidfile)
    {
     cout << "Could not open triangle sides file !" << endl;
     exit(1);
    }
    else
     {
      cout << endl;
      cout << "The triangle sides file is opened" << endl;

      trisidfile >> charSideNo
                 >> charNode1
                 >> charNode2
                 >> charRightTri
                 >> charLeftTri;

      nside = 0;
      do
        {
         nside ++;
         printf("\r");
         cout << "Reading the triangle sides ... " << nside;
         trisidfile >> nside
                    >> TriSides[nside]->Node1
                    >> TriSides[nside]->Node2
                    >> TriSides[nside]->RightTri
                    >> TriSides[nside]->LeftTri;
         //nside ++;
        } while (! trisidfile.eof());
      nside = nside - 1;
     }

// to print the triangle sides data
cout << endl;
cout << endl;

ifstream tri3sidfile("c:\\data\\sercomt$.sid");
  string charTriNo;
  string charSide1;
  string charSide2;
  string charSide3;

   if (! tri3sidfile)
    {
     cout << "Could not open triangles file !" << endl;
     exit(1);
```

```
        }
        else
         {
          cout << endl;
          cout << "The triangles file is opened" << endl;

          tri3sidfile >> charTriNo
                      >> charSide1
                      >> charSide2
                      >> charSide3;

          ntri = 0;
          do
            {
             ntri ++;
             printf("\r");
             cout << "Reading the triangles ... " << ntri;
             tri3sidfile >> ntri
                         >> Tri3Sides[ntri]->Side[0]
                         >> Tri3Sides[ntri]->Side[1]
                         >> Tri3Sides[ntri]->Side[2];
             //ntri ++;
            } while (! tri3sidfile.eof());
          ntri = ntri - 1;
         }

    // to print the three sides
    cout << endl;
    trisidfile.close();
    tri3sidfile.close();
   }

void Contour :: OpenOutputFile()
   {
    char interpolfilename [] = "c:\\data\\sercom10.con";
    interpolfile = fopen(interpolfilename, "w");

    char* charx = "X";
    char* chary = "Y";
    char* charHreq = "Height";
    char* charSegNr = "SegNumber";

     fprintf(interpolfile, "%3s %8s %8s %8s \n", charx, chary, charHreq,
charSegNr);
    if (interpolfile == NULL)
      {
       cerr << "Could not open the interpolation file !" << endl;
       exit(1);
      }
    else
      {
       cout << endl;
       cout << "The file is ready for output" << endl;
      }

   }

void Contour :: AddToFile(float x, float y)
   {
    fprintf(interpolfile, "%3.1f %8.1f %5d %5d \n", x, y, Hreq, SegNr);
   }
```

```
void Contour :: GetMaxMinHeights()
  {
   float x, y, z;

   // Reads the Point file
   ifstream pointfile("c:\\data\\sercomb.xyz");
     if (! pointfile)
       {
        cerr << "Could not open point file !" << endl;
        exit(1);
       }
       else
        {
         cout << endl;
         cout << "The point file is opened" << endl;

         pointfile >> charX >> charY >> charZ;

         npnt = 0;
         pointfile >> x >> y >> z;

         xmin = x;
         xmax = x;

         ymin = y;
         ymax = y;

         Hmax = z;
         Hmin = z;

         do
           {
            npnt ++;
            pointfile >> x >> y >> z;
            if (xmax < x)
                xmax = x;

            if (xmin > x)
                xmin = x;

            if (ymax < y)
                ymax = y;

            if (ymin > y)
                ymin = y;

            if (Hmax < z)
                Hmax = z;
            if (Hmin > z)
                Hmin = z;

            //npnt ++;
           } while (! pointfile.eof());

          rangeH = Hmax - Hmin;
          cout << endl;
          cout << "Max height  : " << Hmax << endl;
          cout << "Min height  : " << Hmin << endl;
          cout << "The H range : " << rangeH << endl;
        }
```

```
   pointfile.close();
  }

void Contour :: GetCoordinate(int s, float& xNd1, float& yNd1,
                                     float& xNd2, float& yNd2,
                                     float& zNd1, float& zNd2)
  {
   int Nd1, Nd2;
   Nd1 = (int)TriSides[s]->Node1;
   Nd2 = (int)TriSides[s]->Node2;

   xNd1 = Point[Nd1]->x;
   yNd1 = Point[Nd1]->y;
   zNd1 = Point[Nd1]->z;

   xNd2 = Point[Nd2]->x;
   yNd2 = Point[Nd2]->y;
   zNd2 = Point[Nd2]->z;
  }

void Contour :: Interpolate(int s, int SegNr, int Hreq)
  {
   float dh, dhi;
   float dx, dy, x, y;

   GetCoordinate(s, xNd1, yNd1, xNd2, yNd2, zNd1, zNd2);

   dh = zNd1 - zNd2;
   if (abs(dh) < accy)
     dh = dh + accy;
   else
     {
      dx = xNd1 - xNd2;
      dy = yNd1 - yNd2;
      dhi = Hreq - zNd1;
      x = xNd1 + ((dx * dhi) / dh);
      y = yNd1 + ((dy * dhi) / dh);

      // to check the x range, dont write the out of the limit x, y
      if ( !(x < xmin) )
       {
        AddToFile(x, y);
        //cout << endl;
        //cout << "side: " << s << endl;
        //cout << "interpolated x, y, Hreq, SegNr: " << endl;
        //cout << x << " " << y << " " << Hreq << " " << SegNr << endl;
        //cout << endl;
       }
     }
  }

bool Contour :: CheckSide(int s)
  {
   int n1, n2;
   float z1, z2;

   n1 = TriSides[s]->Node1;
   n2 = TriSides[s]->Node2;
   z1 = Point[n1]->z;
   z2 = Point[n2]->z;
   z1 = z1 + 0.001;
```

```
    z2 = z2 + 0.001;
    return ((z1 < Hreq) && (z2 > Hreq)) || ((z2 < Hreq) && (z1 > Hreq));
  }

int Contour :: FindFirstTri(int& t, int& s)
  {
    int i;
    bool found;

    //t = 0;
    t = 1;
    found = false;
    while ((t < ntri) && (! found))
      {
       if (! utri[t])
         {
          i = 0;
          //i = 1;
          while ((i < 3) && (! found))
            {
             s = Tri3Sides[t]->Side[i];
             if (CheckSide(s))
                found = true;
             else
                i++;
            }
         }
       if (! found)
          t ++;
      }

    if (! found)
       //t = -1;
       t = 0;

    return t;
  }

void Contour :: FindOtherSide(int t, int s, int& nexts)
  {
    int i;
    bool found;

    found = false;
    i = 0;
    //i = 1;
    while ((i < 3) && (! found))
      {
       nexts = Tri3Sides[t]->Side[i];
       if ((s != nexts) && CheckSide(nexts))
         found = true;
       else
         i++;
      }
  }

void Contour :: FindNextTri(int t, int s, int& nextt)
  {
    if (t == (int)TriSides[s]->RightTri)
      nextt = (int)TriSides[s]->LeftTri;
```

```
    else
      nextt = (int)TriSides[s]->RightTri;
  }

void Contour :: GetContours(int Hreq)
  {
   bool done;
   bool secondpart;
   int startt, starts;
   int s, t;
   float x, y;
   int nextt, nexts;

   //for (int i = 0; i < ntri; i ++)
   for (int i = 1; i < ntri; i ++)
     {
      utri[i] = false;
     }

      do
        {
         //while (FindFirstTri(t, s) != -1)
         while (FindFirstTri(t, s) != 0)
           {
            ii ++;
            SegNr = ii;
            startt = t;
            starts = s;
            done = false;
            secondpart = false;

            do
              {
               utri[t] = true;
               Interpolate(s, SegNr, Hreq);
               FindOtherSide(t, s, nexts);
               FindNextTri(t, nexts, nextt);
               if (nextt == startt)
                 {
                  // found closed contours
                  Interpolate(nexts, SegNr, Hreq);
                  done = true;
                 }
               //else if (nextt == -1)
               else if (nextt == 0)
                      {
                       // hit border
                       Interpolate(nexts, SegNr, Hreq);
                       if (secondpart)
                         done = true;
                       else
                         {
                          FindNextTri(startt, starts, t);
                          //if (t == -1)
                          if (t == 0)
                             done = true;
                          else
                            {
                             s = starts;
                             secondpart = true;
```

```
                                }
                            }
                        }
                    else
                     {
                      t = nextt;
                      s = nexts;
                     }
                  } while (! done);

              }

          //} while ( ! ((t == -1) || (s == -1) || (z != prevH)) ) ;
          } while ( ! ((t == 0) || (s == 0) || (z != prevH)) ) ;
    }

void Contour :: GetInterval(int& interval)
    {
     cout << endl;
     cout << "Enter the required contour interval :" << endl;
     cin >> interval;
    }

void Contour :: MakeContouring()
    {
     prevH = -9999;
     Hmin = Hmin - 0.001;
     Hmax = Hmax + 0.001;
     cout << "No. of triangles :" << ntri << endl;
     cout << "No. of triangle sides :" << nside << endl;
     cout << "No. of nodes :" << npnt << endl;
     cout << endl;
     GetInterval(interval);
     if (interval > rangeH)
       {
        cout << "Sorry, no contours available at the request interval" << endl;
        exit(1);
       }
     else
       {
        Hreq = floor((Hmin / interval) + 1) * interval;
        //cout << "Interpolating the contours ..." << endl;
        //cout << endl;

        while (Hreq < Hmax)
          {
           prevH = Hreq;
           //cout << "Hreq: " << Hreq << endl;
           GetContours(Hreq);
           printf("\r");
           cout << "Interpolating contour height ..." << Hreq;
           Hreq = Hreq + interval;
          }
       }
       cout << endl;
       cout << endl;
       cout << "Contour threading complete." << endl;
       fclose(interpolfile);
    }
```

```
// main program
void main()
  {
   Contour Cont;
   Cont.ii = 0;
   Cont.ReadInputFiles();
   Cont.OpenOutputFile();
   Cont.GetMaxMinHeights();
   Cont.MakeContouring();
   Cont.PressAnyKey();
  }
```

```
// tinwin2d.rh

#define IDI_ICON1 1
#define CM_CLEAR            1125
#define CM_FILEEXIT         1126
#define CM_ABOUT            1127
#define CM_INPUTFILESITEM1     18874 // for XYZ file
#define CM_POPUPITEM1       18873 // for TIN file
```

```
//*************************************************************//
//  A Windows program to display the contours segments         //
//     Input: SegNr file                                       //
//     Output: on screen (using Windows)                       //
//                                                             //
//*************************************************************//

#include <owl/pch.h>
#include <owl/applicat.h>
#include <owl/framewin.h>
#include <owl/decframe.h>
#include <owl/statusba.h>
#include <owl/controlb.h>
#include <owl/buttonga.h>
#include <owl/gdiobjec.h>
#include <owl/chooseco.h>
#include <owl/inputdia.h>
#include <owl/opensave.h>
#include <owl/dc.h>
#include <mem.h>

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>

//#include "plotseg.rh"
#include "tinwin2d.rh"

#define maxtriangle 20000
#define maxpoint 20000

class TSegmentDrawApp : public TApplication
  {
   public:
      TSegmentDrawApp() : TApplication() {}  // call base class constructor
      void InitMainWindow();
  };

class TSegmentWindow : public TWindow
  {
   public:
      TSegmentWindow(TWindow* parent = 0)
         : TWindow(parent) {}

      struct PointStruct
        {
         float x;
         float y;
         float z;
        };
      PointStruct* Point[maxpoint];

      struct SegmentStruct
        {
         float x;
         float y;
         int Hreq;
         int SegNr;
```

```
      };
    SegmentStruct* Segment[maxpoint];

    struct TinStruct
      {
       int Node1;
       int Node2;
       int Node3;
      };
    TinStruct* Tin[maxtriangle];

    FILE* Segmentfile;
    FILE* TINfile;
    FILE* XYZfile;

    TOpenSaveDialog :: TData fileData;

    string charx, chary, charz;
    string charNode1, charNode2, charNode3;
    string charHreq, charSegNr;
    float x, y, z;
    int k;
    int Hreq, SegNr, prevSegNr;
    float xmin, xmax, ymin, ymax;
    float zmin, zmax;
    float xlength, ylength;
    int npnt, ntri, nsegn;
    int MaxX, MaxY;
    float scaleX, scaleY;

    void Transform(int, float&, float&);
    void ReadSegmentFile();
    void ReadXYZTINFile();
    void GetMinMax(float&, float&,
                   float&, float&,
                   float&, float&);
    void Paint(TDC& dc, bool, TRect&);
    void EvSize(uint, TSize&);
    void CmInputFilesItem1();
    void CmPopupItem1();
    void CmFileExit()
         {PostQuitMessage(0);}
    void CmClear();
    void CmAbout();
    DECLARE_RESPONSE_TABLE(TSegmentWindow);
};

DEFINE_RESPONSE_TABLE1(TSegmentWindow, TWindow)
  EV_WM_SIZE,
  EV_COMMAND(CM_INPUTFILESITEM1, CmInputFilesItem1),
  EV_COMMAND(CM_POPUPITEM1, CmPopupItem1),
  EV_COMMAND(CM_ABOUT, CmAbout),
  EV_COMMAND(CM_FILEEXIT, CmFileExit),
  EV_COMMAND(CM_CLEAR, CmClear),
END_RESPONSE_TABLE;


void TSegmentWindow :: CmInputFilesItem1()
  {
   ReadSegmentFile();
```

```
  }

void TSegmentWindow :: CmPopupItem1()
  {
   ReadXYZTINFile();
  }

void TSegmentWindow :: ReadSegmentFile()
  {
   // Allocate memory for segment data
   for (int i = 0; i < maxpoint; i ++)
     {
      Segment[i] = new SegmentStruct;
     }

   // Reads again the the segment file
   //ifstream Segmentfile = "c:\\data\\pent.con";
   ifstream Segmentfile = "c:\\data\\lochgrd4.con";
   //ifstream Segmentfile = "c:\\data\\sercom30.con";
   //ifstream Segmentfile = "c:\\data\\sercom50.con";
   //ifstream Segmentfile = "c:\\data\\sercom10.con";

   if (! Segmentfile)
     {
      MessageBox ("Unable to open file: Contour Segment file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);
     }
   else
     {
      // reads the file header
      Segmentfile >> charx >> chary >> charHreq >> charSegNr;
      nsegn = 0;
        do
         {
          nsegn ++;
          Segmentfile >> Segment[nsegn]->x
                      >> Segment[nsegn]->y
                      >> Segment[nsegn]->Hreq
                      >> Segment[nsegn]->SegNr;
         //nsegn ++;
         } while (! Segmentfile.eof());

     //if (Segmentfile.eof() == true)
           //MessageBox ("Finished read the Segment file", "Segment file
complete",
                  //MB_OK | MB_ICONINFORMATION);
     }
   Segmentfile.close();
  }

void TSegmentWindow :: ReadXYZTINFile()
  {
   // Allocate memory for XYZ data
   for (int k = 0; k < maxpoint; k ++)
     {
      Point[k] = new PointStruct;
     }

   // Reads the XYZ file
   //ifstream XYZfile = "c:\\data\\pent.xyz";
   ifstream XYZfile = "c:\\data\\lochgrd.xyz";
```

```
//ifstream XYZfile = "c:\\data\\sercomb.xyz";


if (!XYZfile)
  {
   MessageBox("Unable to open file: XYZ file",
              "File Error", MB_OK | MB_ICONEXCLAMATION);
  }

else
  {
   // reads the file header
   XYZfile >> charx >> chary >> charz;
   npnt = 0;
   do
     {
      npnt ++;
      XYZfile >> Point[npnt]->x
              >> Point[npnt]->y
              >> Point[npnt]->z;
     } while (! XYZfile.eof());

    //if (XYZfile.eof() == true)
     //MessageBox ("Finished read the XYZ nodes", "XYZ file complete",
                //MB_OK | MB_ICONINFORMATION);
   }

XYZfile.close();


// Allocate memory for Triangle
for (int t = 0; t < maxtriangle; t ++)
  {
   Tin[t] = new TinStruct;
  }

// Reads the TIN file
//ifstream TINfile = "c:\\data\\pent.tin";
ifstream TINfile = "c:\\data\\lochgrd.tin";
//ifstream TINfile = "c:\\data\\sercomt$.tin";

if (!TINfile)
  {
   MessageBox("Unable to open file: TIN file",
              "File Error", MB_OK | MB_ICONEXCLAMATION);
  }

else
  {
   ntri = 0;
   // reads the file header
   TINfile >> charNode1 >> charNode2 >> charNode3;

   do
     {
      ntri ++;
      TINfile >> Tin[ntri]->Node1
              >> Tin[ntri]->Node2
              >> Tin[ntri]->Node3;
     } while (! TINfile.eof());
```

```
        //if (TINfile.eof() == true)
         //MessageBox ("Finished read the TIN nodes", "TIN file complete",
                    //MB_OK | MB_ICONINFORMATION);
       }

   TINfile.close();
  }

void TSegmentWindow :: Transform(int MaxY, float& x, float& y)
  {
   x = (x - xmin) * scaleX;
   y = MaxY - ((y - ymin) * scaleY);
  }

void TSegmentWindow :: CmClear()
  {
   // to clear and redraw
   Invalidate();
  }


void TSegmentWindow :: CmAbout()
  {
   MessageBox ("Contours display with menu - by Alias Abdul-Rahman, 1999",
               "About the Program", MB_OK | MB_ICONINFORMATION);
  }

void TSegmentWindow :: GetMinMax(float& xmax, float& ymax,
                                 float& xmin, float& ymin,
                                 float& xlength, float& ylength)
  {
   // Read the segment file and get the min-max of the coordinates
   //ifstream XYZfile = "c:\\data\\pent.xyz";
   ifstream XYZfile = "c:\\data\\lochgrd.xyz";
   //ifstream XYZfile = "c:\\data\\sercomb.xyz";

   if (! XYZfile)
      MessageBox ("Unable to open file: Segment file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);

   // reads the file header
   XYZfile >> charx >> chary >> charz;
   XYZfile >> x >> y >> z;
      xmin = x;
      xmax = x;

      ymin = y;
      ymax = y;

      do
        {
         XYZfile >> x >> y >> z;

         if (xmin > x)
             xmin = x;

         if (ymin > y)
             ymin = y;

         if (xmax < x)
             xmax = x;
```

```
        if (ymax < y)
            ymax = y;

        } while (! XYZfile.eof());

      xlength = xmax - xmin;
      ylength = ymax - ymin;
  }

void TSegmentWindow :: Paint(TDC& dc, bool, TRect&)
  {
   GetMinMax(xmax, ymax, xmin, ymin, xlength, ylength);

   ReadSegmentFile();      // reads the segment
   ReadXYZTINFile();       // read the TIN

   RECT rect;
   :: GetClientRect(HWindow, &rect);

   scaleX = (rect.right - rect.left) / xlength;
   scaleY = (rect.bottom - rect.top) / ylength;

   int MaxY = (rect.bottom - rect.top);

   for (k = 1; k <= npnt-1; k ++)
     {
      Point[k]->x = (Point[k]->x - xmin) * scaleX;
      Point[k]->y = (rect.bottom - rect.top) - ((Point[k]->y - ymin)
                                                * scaleY);
     }

   // to transform the interpolated points to scale
   for (k = 1; k <= nsegn-1; k ++)
     {
      Segment[k]->x = (Segment[k]->x - xmin) * scaleX;
      Segment[k]->y = (rect.bottom - rect.top) - ((Segment[k]->y - ymin)
                                                * scaleY);
     }

   // set pen style for point display
   /*TPen pen3 = TPen(RGB(255, 0, 255), 2, PS_SOLID);
   SelectObject(dc, pen3);

   for (k = 1; k <= nsegn-1; k ++)
     {
      dc.MoveTo(Segment[k]->x, Segment[k]->y);
      dc.LineTo(Segment[k]->x, Segment[k]->y);
     }
   SelectObject(dc, pen3);
   DeleteObject(pen3);
  */
   // set pen style for TINs display
   TPen pen2 = TPen(RGB(0, 0, 255), 1, PS_SOLID);
   SelectObject(dc, pen2);

   for (k = 1; k <= ntri-1; k ++)
     {
      dc.MoveTo(Point[Tin[k]->Node1]->x,
                Point[Tin[k]->Node1]->y);
      dc.LineTo(Point[Tin[k]->Node2]->x,
                Point[Tin[k]->Node2]->y);
```

```
    dc.LineTo(Point[Tin[k]->Node3]->x,
              Point[Tin[k]->Node3]->y);
    dc.LineTo(Point[Tin[k]->Node1]->x,
              Point[Tin[k]->Node1]->y);
  }
SelectObject(dc, pen2);
DeleteObject(pen2);

// set pen for Segment display
TPen pen1 = TPen(RGB(159, 0, 0), 2, PS_SOLID);

SelectObject(dc, pen1);

float startx, starty;
float nextx, nexty;
int SegNr, nextSegNr, prevSegNr;
float x, y;
int h, nexth;



//ifstream Segmentfile("c:\\data\\pent.con");
ifstream Segmentfile("c:\\data\\lochgrd4.con");
//ifstream Segmentfile("c:\\data\\sercom30.con");
//ifstream Segmentfile("c:\\data\\sercom50.con");
//ifstream Segmentfile("c:\\data\\sercom10.con");

// reads the file header
Segmentfile >> charx >> chary >> charHreq >> charSegNr;
prevSegNr = -999;
Segmentfile >> x >> y >> h >> SegNr;
Transform(MaxY, x, y);
startx = x;
starty = y;

while (! Segmentfile.eof())
   {
    Segmentfile >> nextx >> nexty >> nexth >> nextSegNr;
    Transform(MaxY, nextx, nexty);
    if (SegNr != prevSegNr)
      {
       dc.MoveTo(x, y);
       startx = x;
       starty = y;
      }
    else
      {
       if ( (x == startx) && (y == starty) )
          {
           if (SegNr == nextSegNr)
              dc.MoveTo(x, y);
           else
              dc.LineTo(x, y);
          }
       else
       dc.LineTo(x, y);
      }

    prevSegNr = SegNr;
    SegNr = nextSegNr;
```

```
        x = nextx;
        y = nexty;
        h = nexth;
       }

   //dc.LineTo(x, y);
   dc.MoveTo(x, y);

   Segmentfile.close();
   SelectObject(dc, pen1);
   DeleteObject(pen1);

   // Deallocate memory
   for (int i = 0; i < maxpoint; i ++)
     {
      delete [] Segment[i];
     }

   for (int t = 0; t < maxtriangle; t ++)
     {
      delete [] Tin[t];
     }

   for (int k = 0; k < maxpoint; k ++)
     {
      delete [] Point[k];
     }
  }

// Force the window to repaint if resize
void TSegmentWindow :: EvSize(uint, TSize&)
  {
   :: InvalidateRect(HWindow, 0, true);
  }

void TSegmentDrawApp :: InitMainWindow()
  {
   // construct the decorated MDI frame window
   TDecoratedFrame* frame = new TDecoratedFrame(0,
                     "TIN-to-Contours ver. 1.0", new TSegmentWindow);

   // construct the status bar
   TStatusBar* sb = new TStatusBar(frame, TGadget :: Recessed);

   // construct the control bar
   TControlBar* cb = new TControlBar(frame);

   cb -> Insert(*new TButtonGadget(CM_INPUTFILESITEM1, CM_INPUTFILESITEM1,
                     TButtonGadget :: Command));
   cb -> Insert(*new TButtonGadget(CM_POPUPITEM1, CM_POPUPITEM1,
                     TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_CLEAR, CM_CLEAR,
                     TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_FILEEXIT, CM_FILEEXIT,
                     TButtonGadget :: Command));
```

```
    cb -> Insert(*new TSeparatorGadget);

    cb -> Insert(*new TButtonGadget(CM_ABOUT, CM_ABOUT,
                    TButtonGadget :: Command));

    cb -> SetHintMode(TControlBar :: PressHints);

    // insert the status and control bar into the frame
    frame -> Insert(*sb, TDecoratedFrame :: Bottom);
    frame -> Insert(*cb, TDecoratedFrame :: Top);

    // set the main window and its menu
    SetMainWindow(frame);
    GetMainWindow() -> AssignMenu("COMMANDS");
    EnableCtl3d(true);
   }


int OwlMain(int /*argc*/, char* /*argv*/ [])
   {
    return TSegmentDrawApp().Run();
   }
```

```
#include "tinwin2d.rh"

COMMANDS MENU
{
 POPUP "&InputFiles"
 {
  MENUITEM "&XYZ File", CM_INPUTFILESITEM1
  MENUITEM "&TIN File", CM_POPUPITEM1
 }

 POPUP "&Redraw"
 {
  MENUITEM "&Clear and Redraw", CM_CLEAR
  MENUITEM "E&xit", CM_FILEEXIT
 }

 MENUITEM "&About", CM_ABOUT
}

STRINGTABLE
  {
   CM_INPUTFILESITEM1,   "Get the XYZ file"
   CM_POPUPITEM1,        "Get the TIN file"
   CM_CLEAR,             "Redraw the TINs"
   CM_FILEEXIT,          "Exit the program ..."
   CM_ABOUT,             "About the program ..."
  }

CM_CLEAR BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 91 77 77 77 77 77 77 77 77 99'
 '00 00 79 77 77 77 77 77 77 77 79 17 00 00 77 17'
 '77 87 77 77 77 79 91 77 00 00 77 77 80 08 88 87'
 '77 77 77 77 00 00 77 78 00 00 08 88 77 77 77 77'
 '00 00 77 78 0A 0A A0 08 88 87 77 77 00 00 77 78'
 '0A 0A AA A0 88 88 77 77 00 00 77 80 AA 0A AA AA'
 '00 88 77 77 00 00 77 80 AA A0 AA A0 A0 88 77 77'
 '00 00 77 0A AA A0 AA 0A A0 88 71 17 00 00 77 0A'
 'AA A0 00 AA A0 88 79 97 00 00 78 0A AA A0 AA AA'
 'A0 88 77 77 00 00 78 00 00 00 AA AA A0 88 77 77'
 '00 00 78 88 87 78 0A AA A0 88 77 77 00 00 77 77'
 '77 77 80 AA A0 88 77 77 00 00 77 71 77 77 78 0A'
 'A0 88 77 77 00 00 77 19 77 77 77 80 A0 88 77 77'
 '00 00 77 97 77 77 77 78 00 88 79 77 00 00 71 77'
 '77 77 77 77 88 87 71 19 00 00 19 77 77 77 77 77'
 '77 77 77 71 00 00'
}


CM_FILEEXIT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
```

```
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 EE EE EE EE EE EE EE EE EE EE'
 '00 00 EE EE FF EF EF FE EE EE EE EE 00 00 EE EE'
 'FF EE FE EF EE EE EE EE 00 00 EE EE FF EF FF FF'
 'EE EE EE EE 00 00 EE EE EF EE EF FF EE EE EE EE'
 '00 00 88 88 88 88 88 88 88 88 88 88 00 00 66 66'
 '8A AA A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A A9 A8 77 86 66 66 66 00 00 66 66'
 '8A A9 A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A AA A8 77 86 66 66 66 00 00 66 66'
 '8A A8 87 77 86 66 66 66 00 00 66 66 8A 87 77 77'
 '86 66 66 66 00 00 66 66 88 77 77 77 86 66 66 66'
 '00 00 66 66 88 88 88 88 86 66 66 66 00 00 66 66'
 '66 66 66 66 66 66 66 66 00 00 66 66 66 66 66 66'
 '66 66 66 66 00 00'
}


CM_ABOUT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB B8 88 88 BB BB BB BB'
 '00 00 BB BB BB B8 00 88 BB BB BB BB 00 00 BB BB'
 'BB B8 00 BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00 BB BB BB B8 00 88 BB BB BB BB'
 '00 00 BB BB BB BB 80 08 BB BB BB BB 00 00 BB BB'
 'B8 8B B8 00 8B BB BB BB 00 00 BB BB B8 00 BB 00'
 '8B BB BB BB 00 00 BB BB B8 00 BB 00 8B BB BB BB'
 '00 00 BB BB BB 80 00 00 8B BB BB BB 00 00 BB BB'
 'BB B8 00 00 8B BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00'
}

CM_INPUTFILESITEM1 BITMAP "pointfile.bmp"
/*{
 '42 4D 96 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 18 00 00 00 18 00 00 00 01 00 04 00 00 00'
 '00 00 20 01 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF FF FF FF FF FF FF FF FF'
```

```
 'FF FF FF 00 00 00 00 00 00 00 00 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 0A AA AA AA AA'
 'AA AA AA AA 08 8F FF 0A 99 9A AA AA AA AA AA AA'
 '08 8F FF 0A AA AA AA AA AA AA AA AA 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 00 00 00 00 00'
 '00 00 00 00 08 8F FF 88 88 88 88 88 88 88 88 88'
 '88 8F FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF F0 00 00 00 00'
 '00 00 0F FF FF FF FF F0 FF FF FF FF FF FF 0F FF'
 'FF FF FF F0 F0 00 FF FF FF FF 0F FF FF FF FF F0'
 'FF FF FF F0 00 0F 0F FF FF FF FF F0 FF FF FF FF'
 'FF FF 0F FF FF FF FF F0 FF FF FF 00 0F FF 0F FF'
 'FF FF FF F0 FF FF FF FF FF F0 8F FF FF FF FF F0'
 'F0 00 FF FF FF 08 8F FF FF FF FF F0 FF FF FF FF'
 '00 88 8F FF FF FF FF F0 FF FF FF F0 88 88 8F FF'
 'FF FF FF F0 00 00 00 08 88 88 8F FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF'
}
*/

CM_POPUPITEM1 BITMAP "tinfile.bmp"
/*{
 '42 4D 96 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 18 00 00 00 18 00 00 00 01 00 04 00 00 00'
 '00 00 20 01 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF 00 00 00 00 00 00 00 00 00 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 0A AA AA AA AA'
 'AA AA AA AA 08 8F FF 0A A9 99 9A AA AA AA AA AA'
 '08 8F FF 0A AA AA AA AA AA AA AA AA 08 8F FF 0A'
 'AA AA AA AA AA AA AA AA 08 8F FF 00 00 00 00 00'
 '00 00 00 00 08 8F FF 88 88 88 88 88 88 88 88 88'
 '88 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00'
 '00 00 00 FF FF FF FF FF 0F FF FF FF FF FF F0 FF'
 'FF FF FF FF 0F FF FF FF 00 FF F0 FF FF FF FF FF'
 '0F FF FF FF FF FF F0 FF FF FF FF FF 0F F0 00 0F'
 'FF FF F0 FF FF FF FF FF 0F FF FF FF FF FF F0 FF'
 'FF FF FF FF 0F FF FF F0 0F FF F0 FF FF FF FF FF'
 '0F FF FF FF FF FF 08 FF FF FF FF FF 0F F0 00 FF'
 'FF F0 88 FF FF FF FF FF 0F FF FF FF F0 08 88 FF'
 'FF FF FF FF 0F FF FF FF 08 88 88 FF FF FF FF FF'
 '00 00 00 00 88 88 88 FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF'
}

*/
```

3D Objects Heights-to-Terrain Interpolation

```
//*********************************************************************//
// Project: Height interpolation from Roof to DTM (TIN)                //
// File: heightinterpol.cpp                                            //
// Input:  .XYZ file, .TIN file, and ROOF file                         //
// Output:  Interpolated Z file                                        //
//                                                                     //
//*********************************************************************//

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <mem.h>
#include <stdio.h>
#include <string>
#include <conio.h>
#include <math.h>

#define maxpoint 7000
#define maxtriangle 20000
#define accy 1.0e-10

#define max(a, b) ( ((a) > (b)) ? (a) : (b) )
#define min(a, b) ( ((a) < (b)) ? (a) : (b) )

#define max3(a, b, c) ((a) > (b) ? ((a)>(c)?(a):(c)) : ((b)>(c)?(b):(c)))
#define min3(a, b, c) ((a) < (b) ? ((a)<(c)?(a):(c)) : ((b)<(c)?(b):(c)))

class Interpolation
  {
   public:
     struct Point
       {
        float x, y, z;
       };

     struct TIN
       {
        int Node1, Node2, Node3;
       };

     Point* pnt[maxpoint];
     TIN* tin[maxtriangle];


     struct ROOF
       {
        float xr;
        float yr;
        float zr;
       };
     ROOF* roof[maxpoint];
```

```
  typedef struct InterfileStruct
    {
     float xi;
     float yi;
     float zi;
    } Interfile;
  Interfile Element;

  Interpolation() {}
  ~Interpolation() {}

  void ReadXYZ();
  void ReadTIN();
  void ReadROOF();
  void Get3Nodes(int, float&, float&,
                      float&, float&,
                      float&, float&);
  void GetSign(float, float,
               float, float, float, float, float&);
  int FindTri(double, double);
  int PntInTri(double, double,
               double, double,
               double, double,
               double, double);
  void PointInTriangleTest(float, float, bool&);
  void GetPlane(float, float, float, float,
                float, float, float, float, float,
                float&, float&, float&, float&);
  void GetHeightIntersect(float, float, float, float,
                                  float, float, float&);
  float ZInTri(float, float,
               float, float, float,
               float, float, float,
               float, float, float);
  float Rad(float);
  float Deg(float);
  float Max3(float a, float b, float c);
  float Min3(float a, float b, float c);
  int CheckQuadrant(float, float);
  float Azimuth(float, float, float, float);
  void GetAngle(float, float, float, float,
                float, float, float&);
  float Bearing(float, float, float, float);
  void MakeInterpolation();
  void GetIntersectFile();
  void AddZtoFile(Interfile);
  void DeallocateMemory();
  void PressAnyKey();

private:
  int npnt;
  int ntin, t;
  int rpnt, maxroofpnt;
  int Nd1, Nd2, Nd3;
  int Node1, Node2, Node3;
  float xNd1, xNd2, xNd3;
  float yNd1, yNd2, yNd3;
  float zNd1, zNd2, zNd3;
  float zint;
```

```
    bool intri;
    FILE* INTERSECTfile;
    float xroof, yroof, zroof;
    //float xi, yi, zi;
    float a, b, c, d;
};


void Interpolation :: DeallocateMemory()
  {
   // deallocate memory for XYZ table
   for (int i = 0; i < maxpoint; i ++)
     {
      delete [] pnt[i];
     }

   // deallocate memory for TIN table
   for (int tt = 0; tt < maxtriangle; tt ++)
     {
      delete [] tin[tt];
     }


   // deallocate memory for ROOF table
   for (int r = 0; r < maxpoint; r ++)
     {
      delete [] roof[r];
     }

  }


void Interpolation :: ReadXYZ()
  {
   ifstream XYZfile = "c:\\data\\lochgrd.xyz";

   if (! XYZfile)
      cout << "Unable to open file: XYZ file" << endl;


   // allocate memory for XYZ table
   for (int i =  0; i < maxpoint; i ++)
     {
      pnt[i] = new Point;
     }

   //reads the file header
   string charX, charY, charZ;
   XYZfile >> charX >> charY >> charZ;

   npnt = 0;
   do
     {
      npnt ++;
      XYZfile >> pnt[npnt] -> x
              >> pnt[npnt] -> y
              >> pnt[npnt] -> z;
     } while (! XYZfile.eof());

   XYZfile.close();
  }
```

```
void Interpolation :: ReadTIN()
  {
   ifstream TINfile = "c:\\data\\lochgrd.tin";

   string charNode1, charNode2, charNode3;

   if (! TINfile)
      cout << "Unable to open file: TIN file" << endl;

      // allocate memory for TIN table
      for (int t = 0; t < maxtriangle; t ++)
       {
        tin[t] = new TIN;
       }

      TINfile >> charNode1 >> charNode2 >> charNode3;

      ntin = 0;
      do
        {
         ntin ++;
         TINfile >> tin[ntin] -> Node1
                 >> tin[ntin] -> Node2
                 >> tin[ntin] -> Node3;

        } while (! TINfile.eof());

   TINfile.close();
  }

void Interpolation :: ReadROOF()
  {
   ifstream ROOFfile = "c:\\data\\build3.xyz";

   if (! ROOFfile)
      cout << "Unable to open file: Roof file" << endl;

   // allocate memory for ROOF table
   for (int r =  0; r < maxpoint; r ++)
     {
      roof[r] = new ROOF;
     }

   //reads the file header
   string charX, charY, charZ;
   ROOFfile >> charX >> charY >> charZ;

   rpnt = 0;
   do
     {
      rpnt ++;
      ROOFfile >> roof[rpnt] -> xr
               >> roof[rpnt] -> yr
               >> roof[rpnt] -> zr;
     } while (! ROOFfile.eof());

   maxroofpnt = rpnt;

   ROOFfile.close();
  }
```

```
void Interpolation :: GetIntersectFile()
  {
   char intersectfilename [] = "c:\\data\\lochr2t.xyz";
   INTERSECTfile = fopen(intersectfilename, "w");

   char* charX = "X";
   char* charY = "Y";
   char* charZ = "Z";

   fprintf(INTERSECTfile, "%3s %5s %8s \n", charX, charY, charZ);

   if (INTERSECTfile == NULL)
     {
      cerr << "Could not open INTERSECT file !" << endl;
      exit(1);
     }
   else
     {
      cout << endl;
      cout << "The INTERSECT file is okay for writing." << endl;
     }
  }

void Interpolation :: AddZtoFile(Interfile Int)
  {
   fprintf(INTERSECTfile, "%5.1f %5.1f %5.1f \n", Int.xi, Int.yi, Int.zi);
  }

float Interpolation :: Max3(float a, float b, float c)
  {
   float max;

   max = max(a, b);
   max = max(max, c);
   return max;
  }

float Interpolation :: Min3(float a, float b, float c)
  {
   float min;

   min = min(a, b);
   min = min(min, c);
   return min;
  }

float Interpolation :: Rad(float Deg)
  {
   return M_PI * Deg / 180.0;
  }

float Interpolation :: Deg(float Radian)
  {
   return Radian * 180.0 / M_PI;
  }

float Interpolation :: Bearing(float x1, float y1, float x2, float y2)
  {
   float theta;
```

```
    if (abs(y2 - y1) < accy)
      theta = M_PI / 2;
    else
      theta = atan((x2 - x1) / (y2 - y1));

    return theta;
  }

float Interpolation :: Azimuth(float x1, float y1, float x2, float y2)
  {
    float theta;
    float dx, dy;
    int quadrant;

    dx = x2 - x1;
    dy = y2 - y1;
    if ((dx == 0) && (dy > 0))
      theta = 0;
    else
      if ((dx > 0) && (dy == 0))
        theta = M_PI / 2;
    else
      if ((dx == 0) && (dy < 0))
        theta = M_PI;
    else
      if ((dx < 0) && (dy == 0))
        theta = 3 * M_PI / 2;
    else
      {
       quadrant = CheckQuadrant(dx, dy);

       switch (quadrant)
         {
          case 1:
            theta = Bearing(x1, y1, x2, y2);
            break;

          case 2:
            theta = (2 * M_PI) - abs(Bearing(x1, y1, x2, y2));
            break;

          case 3:
            theta = M_PI + abs(Bearing(x1, y1, x2, y2));
            break;

          case 4:
            theta = M_PI - abs(Bearing(x1, y1, x2, y2));
            break;
         }
      }
    return theta;

  }

void Interpolation :: GetAngle(float xa, float ya, float xb, float yb,
                               float xc, float yc, float& angle)
  {
    bool minus;
    float Az1, Az2;
```

```
    minus = false;
    Az1 = Azimuth(xb, yb, xa, ya);
    Az2 = Azimuth(xb, yb, xc, yc);
    angle = Az2 - Az1;
    if (angle < 0)
      minus = true;
    if (abs(angle) > M_PI)
      {
       angle = (2 * M_PI) - abs(angle);
       if ( ! (minus))
         angle = -angle;  // change direction
      }
  }

void Interpolation :: Get3Nodes(int i, float& xNd1, float& yNd1,
                                       float& xNd2, float& yNd2,
                                       float& xNd3, float& yNd3)
  {
   Nd1 = tin[i]->Node1;
   Nd2 = tin[i]->Node2;
   Nd3 = tin[i]->Node3;

   xNd1 = pnt[Nd1]->x;  // (x, y, z) Node1
   yNd1 = pnt[Nd1]->y;
   zNd1 = pnt[Nd1]->z;

   xNd2 = pnt[Nd2]->x;  // (x, y, z) Node2
   yNd2 = pnt[Nd2]->y;
   zNd2 = pnt[Nd2]->z;

   xNd3 = pnt[Nd3]->x;  // (x, y, z) Node3
   yNd3 = pnt[Nd3]->y;
   zNd3 = pnt[Nd3]->z;
  }

int Interpolation :: CheckQuadrant(float dx, float dy)
  {
   int c;

   if ((dx > 0) && (dy > 0))
     c = 1;
   else
   if ((dx < 0) && (dy > 0))
     c = 2;
   else
   if ((dx < 0) && (dy < 0))
     c = 3;
   if ((dx > 0) && (dy < 0))
     c = 4;

   return c;

  }


void Interpolation :: GetSign(float xr, float yr,
                              float x1, float y1, float x2, float y2, float&
sign)
  {
   float x21, y21;
```

```
    float rsq, rinv;
    float a, b, c;

    x21 = x2 - x1;
    y21 = y2 - y2;
    rsq = (x21 * x21) + (y21 * y21);
    if (rsq < accy)
      cout << "The points coincide" << endl;
    else
      rinv = 1.0 / sqrt(rsq);
      a = -(y21) * rinv;
      b = x21 * rinv;
      c = ((x1*y2) - (x2* y1)) * rinv;
      sign = (a*xr) + (b*yr) + c;
  }


int Interpolation :: PntInTri(double x, double y,
                              double x1, double y1,
                              double x2, double y2,
                              double x3, double y3)
  {
   double o12, o23, o31;
   if (x > max3(x1, x2, x3))
      return 0;
   if (x < min3(x1, x2, x3))
      return 0;
   if (y > max3(y1, y2, y3))
      return 0;
   if (y < min3(y1, y2, y3))
      return 0;

   o12 = x1*y2 - x2*y1 + x2*y - x*y2 + x*y1 - x1*y;
   o23 = x2*y3 - x3*y2 + x3*y - x*y3 + x*y2 - x2*y;
   o31 = x3*y1 - x1*y3 + x1*y - x*y1 + x*y3 - x3*y;

   if      (o12 > 0) return o23 >= 0 && o31 >= 0;
   else if (o12 < 0) return o23 <= 0 && o31 <= 0;
   else
    {
     if      (o23 > 0) return o31 >= 0;
     else if (o23 < 0) return o31 <= 0;
     else              return 1;
    }
  }


int Interpolation :: FindTri(double x, double y)
  {
   int Node1, Node2, Node3, i, found;
   i = 1;
   found = 0;
   while (i <= ntin && !found)
     {
      Node1 = tin[i]->Node1;
      Node2 = tin[i]->Node2;
      Node3 = tin[i]->Node3;
      found = PntInTri(x, y,
                       xNd1, yNd1,
                       xNd2, yNd2,
```

```
                        xNd3, yNd3);
      if (!found)
      i ++;
     }
    return i;
  }

void Interpolation :: PointInTriangleTest(float xr, float yr, bool& intri)
  {
   float angle1, angle2, angle3;
   float SumAng;

   if ((xr == xNd1) && (yr == yNd1))
     {
      zint = zNd1;
      intri = true;
     }
   else
   if ((xr == xNd2) && (yr == yNd2))
     {
      zint = zNd2;
      intri = true;
     }
   else
   if ((xr == xNd3) && (yr == yNd3))
     {
      zint = zNd3;
      intri = true;
     }
   else
     {
      GetAngle(xNd1, yNd1, xr, yr, xNd2, yNd2, angle1);
      GetAngle(xNd2, yNd2, xr, yr, xNd3, yNd3, angle2);
      GetAngle(xNd3, yNd3, xr, yr, xNd1, yNd1, angle3);
      SumAng = angle1 + angle2 + angle3;
      cout << "Sum Angle : " << SumAng << endl;

      if (abs(int(Deg(SumAng))) >= 360)
        intri = true;
      else
      if (int(Deg(SumAng)) == 0)
        intri = false;
      else
        {
         cout << "error ..." << endl;
         cout << "x1: " << xNd1 << " " << "y1: " << yNd1 << " "
              << "Angle1: " << Deg(angle1) << endl;
         cout << "x2: " << xNd2 << " " << "y2: " << yNd2 << " "
              << "Angle2: " << Deg(angle2) << endl;
         cout << "x3: " << xNd3 << " " << "y3: " << yNd3 << " "
              << "Angle3: " << Deg(angle3) << endl;
         cout << "xr: " << xr << " " << "yr: " << yr << " "
              << "Sum Angle: " << Deg(SumAng) << endl;
        }
     }

  }

void Interpolation :: GetPlane(float xj, float yj,
                               float xk, float yk,
```

```
                                float xl, float yl,
                                float zj, float zk, float zl,
                                float& a, float& b, float& c, float& d)
  {
   float xkj, ykj, zkj;
   float xlj, ylj, zlj;

   xkj = xk - xj;
   ykj = yk - yj;
   zkj = zk - zj;
   xlj = xl - xj;
   ylj = yl - yj;
   zlj = zl - zj;

   a = (ykj * zlj) - (zkj * ylj);
   b = (zkj * xlj) - (xkj * zlj);
   c = (xkj * ylj) - (ykj * xlj);
   d = -((xk * a) + (yk * b) + (zk * c));
  }


float Interpolation :: ZInTri(float x, float y,
                              float x1, float y1, float z1,
                              float x2, float y2, float z2,
                              float x3, float y3, float z3)
  {
   float dx, dy, dx2, dy2, dz2, dx3, dy3, dz3, denom;
   dx = x - x1;
   dy = y - y1;
   dx2 = x2 - x1;
   dy2 = y2 - y1;
   dz2 = z2 - z1;
   dx3 = x3 - x1;
   dy3 = y3 - y1;
   dz3 = z3 - z1;
   denom = dx2 * dy3 - dx3 * dy2;
   if (fabs(denom) < 1e-10)
   return z1;

   return z1 - (dx * (dy2*dz3-dy3*dz2) - dy * (dx2*dz3 - dx3*dz2)) / denom;
  }


void Interpolation :: GetHeightIntersect(float xk, float yk, float a, float
b,
                                         float c, float d, float& z)
  {
   float xl, yl, zk, zl, f, g, h, t;
   float denom;
   float x, y;

   xl = xk;
   yl = yk;
   zk = 0;
   zl = 32767;  // false height
   f = xl - xk;
   g = yl - yk;
   h = zl - zk;
   denom = (a*f) + (b*g) + (c*h);
```

```
    if (fabs(denom) < accy)
      cout << "lines and planes are parallel ..." << endl;
    else
      {
       t = -(a*xk + b*yk + c*zk + d) / denom;
       x = xk + (f*t);
       y = yk + (g*t);
       z = zk + (h*t);  // this is the intersected Z and
                        // the triangle plane surface
      }
  }

void Interpolation :: MakeInterpolation()
  {
   //ifstream ROOFfile = "c:\\data\\build2.xyz";

   ReadXYZ();
   ReadTIN();
   ReadROOF();
   GetIntersectFile();

   cout << "Interpolating ....." << endl;
   cout << endl;

   intri = false;
   for (int r = 1; r <= maxroofpnt; r ++)
     {
      xroof = roof[r]->xr;
      yroof = roof[r]->yr;
      //int k = 1;

      for (int k = 1; k < ntin; k ++)
        {
         Get3Nodes(k, xNd1, yNd1, xNd2, yNd2, xNd3, yNd3);

         if ( (xroof >= Min3(xNd1, xNd2, xNd3)) &&
              (xroof <= Max3(xNd1, xNd2, xNd3)) &&
              (yroof >= Min3(yNd1, yNd2, yNd3)) &&
              (yroof <= Max3(yNd1, yNd2, yNd3)) )
            {
             cout << "roof pnt : " << rpnt << endl;
             PointInTriangleTest(xroof, yroof, intri);

             /*t = FindTri(xroof, yroof);
             if (t > ntin)
               zroof = -999;
             else
               {
                Node1 = tin[t]->Node1;
                Node2 = tin[t]->Node2;
                Node3 = tin[t]->Node3;
                zint = ZInTri(xroof, yroof, xNd1, yNd1, zNd1,
                                            xNd2, yNd2, zNd2,
                                            xNd3, yNd3, zNd3);

                Element.xi = xroof;
                Element.yi = yroof;
                Element.zi = zint;
                AddZtoFile(Element);
               }
             */
```

```
            if (intri == true)
              {
               GetPlane(xNd1, yNd1,
                        xNd2, yNd2,
                        xNd3, yNd3,
                        zNd1, zNd2, zNd3,
                        a, b, c, d);
               zint = ZInTri(xroof, yroof, xNd1, yNd1, zNd1,
                                           xNd2, yNd2, zNd2,
                                           xNd3, yNd3, zNd3);
               Element.xi = xroof;
               Element.yi = yroof;
               Element.zi = zint;
               AddZtoFile(Element);
              }
            }
        }
      } // the for loop

   fclose(INTERSECTfile);
  }

void Interpolation :: PressAnyKey()
  {
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

// main program
void main()
  {
   Interpolation Inter;
   Inter.MakeInterpolation();
   Inter.PressAnyKey();
  }
```

Line Segment-to-Arc/Info Conversion Code

```
//*********************************************************//
//                                                         //
//   A Program to convert SEG file to LIN file (Arc/Info)  //
//      Input: SEG file                                    //
//      Output: LIN file                                   //
//                                                         //
//*********************************************************//

#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream.h>
#include <conio.h>
#include <math.h>
#include <string>

#define maxpoint 7000

class Seg2Arc
  {
   public:
     float x, y;
     float startx, starty;
     float nextx, nexty;
     int z, nextz, startz;
     int npnt;
     int SegNr, prevSegNr, nextSegNr, startSegNr;

     void MakeArc();

     Seg2Arc(){}
     ~Seg2Arc(){}
  };

// the implementation
void Seg2Arc :: MakeArc()
  {
   string charX, charY, charZ, charSegNr;
   char* charEND = "END";

   ifstream Segmentfile("c:\\data\\cser120.con");
   char Arcfilename [] = "c:\\data\\cser120.lin";

   FILE* Arcfile = fopen(Arcfilename, "w");

   // reads the file header
   Segmentfile >> charX >> charY >> charZ >> charSegNr;
   npnt = 1;
   //prevSegNr = -999;
   Segmentfile >> x >> y >> z >> SegNr;
   startx = x;
```

```
   starty = y;
   startz = z;
   prevSegNr = SegNr;

   while (! Segmentfile.eof())
     {
      printf("\r");
      cout << "Converting to LIN format ...." << npnt;
      Segmentfile >> nextx >> nexty >> nextz >> nextSegNr;

      if (SegNr != prevSegNr)
        {
         fprintf(Arcfile, "%s \n", charEND);
         fprintf(Arcfile, "%5d \n", nextz);
         fprintf(Arcfile, "%3.1f   %3.1f \n", x, y);
         startx = x;
         starty = y;
        }
      else
        {
         if ((x == startx) && (y == starty))
           {
            if (SegNr == prevSegNr)
              {
               fprintf(Arcfile, "%s \n", charEND);
               fprintf(Arcfile, "%5d \n", z);
               fprintf(Arcfile, "%3.1f   %3.1f \n", x, y);
              }
            else
              fprintf(Arcfile, "%3.1f   %3.1f \n", x, y);
           }
         else
         fprintf(Arcfile, "%3.1f   %3.1f \n", x, y);
        }

      prevSegNr = SegNr;
      SegNr = nextSegNr;
      x = nextx;
      y = nexty;
      z = nextz;
      npnt ++;
     }
    fprintf(Arcfile, "%s \n", charEND);
    fprintf(Arcfile, "%s \n", charEND);

    Segmentfile.close();
    fclose(Arcfile);
  }


void main()
   {
    Seg2Arc Arc;
    Arc.MakeArc();
   }
```

3D Viewing Code

```
//*********************************************************//
// A Program to View TIN with arcs and edges on screen     //
// A Program to Display 3D TINs Using OWL                  //
// Input    : (.XYZ), (.TIN), and (.ARC) files             //
// Output   : Windows display                              //
//                                                         //
//*********************************************************//

#include <owl/pch.h>
#include <owl/applicat.h>
#include <owl/framewin.h>
#include <owl/dc.h>
#include <owl/decframe.h>
#include <owl/statusba.h>
#include <owl/controlb.h>
#include <owl/buttonga.h>
#include <owl/gdiobjec.h>
#include <owl/chooseco.h>
#include <owl/inputdia.h>
#include <owl/opensave.h>

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <math.h>
#include <mem.h>
#include <string.h>

#include "view3d.rh"


// constants for maxpoints, tins, and arcs
const int maxtin = 2000;
const int maxpoint = 700;
const int maxarc = 200;
const int maxsegment = 1000;

// view parameters constants

const double theta = -60.0;  // was -60.0
const double phi = 60.0;
const double ViewPlaneDist = 2000.0; // 500 for the trylake
                                     // the bigger the ViewPlaneDist
                                     // -> larger view
const double rho = 2500.0;           // 1500 for the trylake

class TView3DWindowApp : public TApplication
  {
   public:
```

```
    TView3DWindowApp() : TApplication() {}  // call base class constructor
    void InitMainWindow();
};


class TView3D : public TWindow
  {
   public:
    TView3D(TWindow* parent = 0)
            : TWindow(parent) {}

    typedef double Matrix[4][4];
    struct Vector
      {
       double  x, y, z;
      };

    struct Point
      {
       double x, y, z;
      };

    struct Pointt
      {
       float x, y, z;
      };

    struct TIN
      {
       int Node1, Node2, Node3;
      };

    struct TIN2
      {
       int Node1, Node2, Node3;
      };

    Point* pnt[maxpoint];
    Pointt* pntt[maxpoint];
    TIN* tin[maxtin];
    TIN2* tinn[maxtin];

    struct Polygon
      {
       int Snode;
       int Enode;
      };
    Polygon* poly_ptr[maxarc];

    struct RoofArc
      {
       int Snode;
       int Enode;
      };
    RoofArc* roofarc[maxarc];

    FILE* ARCfile;
    FILE* ARCRooffile;

    // struct for trees
```

```
struct TreePoints
  {
   double x;
   double y;
   double z;
  };
TreePoints* treepnts[maxpoint];

struct TreeArcs
  {
   int Snode;
   int Enode;
  };
TreeArcs* treearc[maxarc];

// file for the tree points and arcs
FILE* Treepntfile;
FILE* Treearcfile;

struct Segment
  {
   double x;
   double y;
   int Hreq;
   int SegNr;
  };
Segment* Seg[maxsegment];

int i, t, tt, k;
int npnt, ntin, narc, n;
int npntt, ntinn, narcr;
int ntree, narct;
string charNode1, charNode2, charNode3;
double xmin, xmax, ymin, ymax;
double zmin, zmax;
double xlength, ylength, zlength;

double startx, starty;
double nextx, nexty;
int SegNr, nextSegNr, prevSegNr;
double x, y;
int h, nexth;

double MaxX, MaxY;
double xmid, ymid;
double screen_x, screen_y;
double scaleX, scaleY, scaleZ;
double sin_theta;
double cos_theta;
double sin_phi;
double cos_phi;
double x_view, y_view, z_view;
Matrix viewT;
Vector VecWorld, VecView;
double DotProduct(Vector, Vector);
double Magnitude(Vector);
void Transform(int, float&, float&);
void Normalize(Vector, Vector&);
void SetZeroMatrix(Matrix&);
void SetIdentityMatrix(Matrix&);
```

```
      void VectorMatrix(Vector, Matrix, Vector&);
      void SetViewTransformationMatrix(Matrix);
      void SetViewVariables(double&, double&, double&);
      void ReadXYZ();
      void ReadXYZ2();
      void ReadTIN();
      void ReadTIN2();
      void ReadARC();
      void ReadRoofArc();
      void ReadTreeXYZ();
      void ReadTreeArc();
      void ReadSegment();
      void DeallocateMemory();
      void GetMinMax(double&, double&, double&, double&, double&, double&,
                     double&, double&, double&);
      void Perspective(double, double, double, double&, double&);
      double InRadians(double);
      void Paint(TDC& dc, bool, TRect&);
      void EvSize(uint, TSize&);
      void CmInputFilesItem1();
      void CmPopupItem1();
      void CmFileExit()
           {PostQuitMessage(0);}
      void CmClear();
      void CmAbout();

      ~TView3D() {}

      DECLARE_RESPONSE_TABLE(TView3D);
   };

DEFINE_RESPONSE_TABLE1(TView3D, TWindow)
   EV_WM_SIZE,
   EV_COMMAND(CM_INPUTFILESITEM1, CmInputFilesItem1),
   EV_COMMAND(CM_POPUPITEM1, CmPopupItem1),
   EV_COMMAND(CM_ABOUT, CmAbout),
   EV_COMMAND(CM_FILEEXIT, CmFileExit),
   EV_COMMAND(CM_CLEAR, CmClear),
END_RESPONSE_TABLE;


// the definitions section
void TView3D :: DeallocateMemory()
  {
   // for XYZ table
   for (i = 0; i < maxpoint; i ++)
     {
      delete [] pnt[i];
     }

   // for XYZ2 table
   for (int ii = 0; ii < maxpoint; ii ++)
     {
      delete [] pntt[ii];
     }

   // for TIN table
   for (t = 0; t < maxtin; t ++)
     {
      delete [] tin[t];
```

```
    }

  // for TIN2 table
  for (int tt = 0; tt < maxtin; tt ++)
    {
     delete [] tinn[tt];
    }

  for (int ii = 0; ii < maxarc; ii ++)
    {
     delete [] poly_ptr[ii];
    }

  for (int ir = 0; ir < maxarc; ir ++)
    {
     delete [] roofarc[ir];
    }

  // for the tree points
  for (int tp = 0; tp < maxpoint; tp ++)
    {
     delete [] treepnts[tp];
    }

  // for the tree arcs
  for (int ta = 0; ta < maxarc; ta ++)
    {
     delete [] treearc[ta];
    }

  // for the segment contours
  for (int sg = 0; sg < maxsegment; sg ++)
    {
     delete [] Seg[sg];
    }

 }

void TView3D :: ReadXYZ()
 {
  //ifstream XYZfile = "c:\\data\\lakebore.xyz";
  //ifstream XYZfile = "c:\\data\\test.xyz";
  ifstream XYZfile = "c:\\data\\lochass.xyz";

  if (! XYZfile)
     MessageBox ("Unable to open file: XYZ file", "File Error",
                    MB_OK | MB_ICONEXCLAMATION);

  // for XYZ table
  for (i =  0; i < maxpoint; i ++)
    {
     pnt[i] = new Point;
    }

  //reads the file header
  string charX, charY, charZ;
  XYZfile >> charX >> charY >> charZ;

  npnt = 0;
  do
    {
```

```
      npnt ++;
      XYZfile >> pnt[npnt] -> x
              >> pnt[npnt] -> y
              >> pnt[npnt] -> z;
     } while (! XYZfile.eof());

   XYZfile.close();
  }

void TView3D :: ReadXYZ2()
  {
   //ifstream XYZ2file = "c:\\data\\test400.xyz";
   //ifstream XYZ2file = "c:\\data\\lake200.xyz";
   //ifstream XYZ2file = "c:\\data\\testrnf.xyz";
   ifstream XYZ2file = "c:\\data\\lochfpr.xyz";

   if (! XYZ2file)
      MessageBox ("Unable to open file: XYZ file", "File Error",
                    MB_OK | MB_ICONEXCLAMATION);

   // for XYZ table
   for (i =  0; i < maxpoint; i ++)
     {
      pntt[i] = new Pointt;
     }

   //reads the file header
   string charX, charY, charZ;
   XYZ2file >> charX >> charY >> charZ;

   npntt = 0;
   do
     {
      npntt ++;
      XYZ2file >> pntt[npntt] -> x
               >> pntt[npntt] -> y
               >> pntt[npntt] -> z;
     } while (! XYZ2file.eof());

   XYZ2file.close();
  }



void TView3D :: ReadTIN()
  {
   string charNode1, charNode2, charNode3;

   //ifstream TINfile = "c:\\data\\lakebo.tin";
   //ifstream TINfile = "c:\\data\\testr.tin";
   ifstream TINfile = "c:\\data\\lochas.tin";

   if (! TINfile)
      MessageBox("Unable to open file: TIN file",
                 "File Error", MB_OK | MB_ICONEXCLAMATION);

      // for TIN table
      for (t = 0; t < maxtin; t ++)
       {
        tin[t] = new TIN;
       }
```

```
      TINfile >> charNode1 >> charNode2 >> charNode3;
      ntin = 0;
      do
        {
         ntin ++;
         TINfile >> tin[ntin] -> Node1
                 >> tin[ntin] -> Node2
                 >> tin[ntin] -> Node3;

        } while (! TINfile.eof());

   TINfile.close();
  }

void TView3D :: ReadTIN2()
  {
   //ifstream TIN2file = "c:\\data\\linetin.tin";
   ifstream TIN2file = "c:\\data\\lake2ft.tin";

   if (! TIN2file)
      MessageBox("Unable to open file: TIN file",
                 "File Error", MB_OK | MB_ICONEXCLAMATION);

      // for TIN2 table
      for (tt = 0; tt < maxtin; tt ++)
       {
        tinn[tt] = new TIN2;
       }

      TIN2file >> charNode1 >> charNode2 >> charNode3;
      ntinn = 0;
      do
        {
         ntinn ++;
         TIN2file >> tinn[ntinn] -> Node1
                  >> tinn[ntinn] -> Node2
                  >> tinn[ntinn] -> Node3;

        } while (! TIN2file.eof());

   TIN2file.close();
  }



void TView3D :: ReadARC()
  {
   string charSnode, charEnode;

   // Reads again the the XYZ file
   //ifstream ARCfile = "c:\\data\\lake.arc";
   //ifstream ARCfile = "c:\\data\\test.arc";
   ifstream ARCfile = "c:\\data\\lake.arc";

   // Allocate memory for Arcs
   for (int ii = 0; ii < maxarc; ii ++)
     {
      poly_ptr[ii] = new Polygon;
     }

   if (! ARCfile)
```

```
    {
     MessageBox ("Unable to open file: ARC file", "FileError",
                 MB_OK | MB_ICONEXCLAMATION);
    }
  else
    {
     // reads the file header
     ARCfile >> charSnode >> charEnode;
     narc = 0;
       do
        {
         narc ++;
         ARCfile >> poly_ptr[narc] -> Snode
                 >> poly_ptr[narc] -> Enode;

        } while (! ARCfile.eof());

    //if (ARCfile.eof() == true)
       //MessageBox ("Finished read the ARC points", "ARC file complete",
                   //MB_OK | MB_ICONINFORMATION);
    }
  ARCfile.close();
 }

void TView3D :: ReadRoofArc()
 {
  string charSnode, charEnode;

  //ifstream ARCRooffile = "c:\\data\\roofarc.arc";
  ifstream ARCRooffile = "c:\\data\\lochbld.arc";

  // Allocate memory for Arcs
  for (int ii = 0; ii < maxarc; ii ++)
    {
     roofarc[ii] = new RoofArc;
    }

  if (! ARCRooffile)
    {
     MessageBox ("Unable to open file: ARC roof file", "FileError",
                 MB_OK | MB_ICONEXCLAMATION);
    }
  else
    {
     // reads the file header
     ARCRooffile >> charSnode >> charEnode;
     narcr = 0;
       do
        {
         narcr ++;
         ARCRooffile >> roofarc[narcr] -> Snode
                     >> roofarc[narcr] -> Enode;

        } while (! ARCRooffile.eof());

    //if (ARCfile.eof() == true)
       //MessageBox ("Finished read the ARC points", "ARC file complete",
                   //MB_OK | MB_ICONINFORMATION);
    }
  ARCRooffile.close();
 }
```

```
// for the tree points
void TView3D :: ReadTreeXYZ()
  {
   ifstream Treepntfile = "c:\\data\\tree1.xyz";

   if (! Treepntfile)
      MessageBox ("Unable to open file: XYZ file", "File Error",
                    MB_OK | MB_ICONEXCLAMATION);

   // for tree XYZ table
   for (i =  0; i < maxpoint; i ++)
     {
      treepnts[i] = new TreePoints;
     }

   //reads the file header
   string charX, charY, charZ;
   Treepntfile >> charX >> charY >> charZ;

   ntree = 0;
   do
     {
      ntree ++;
      Treepntfile >> treepnts[ntree] -> x
                  >> treepnts[ntree] -> y
                  >> treepnts[ntree] -> z;
     } while (! Treepntfile.eof());

   Treepntfile.close();
  }

void TView3D :: ReadTreeArc()
  {
   string charSnode, charEnode;

   ifstream Treearcfile = "c:\\data\\tree1.arc";

   // Allocate memory for tree arcs
   for (int ii = 0; ii < maxarc; ii ++)
     {
      treearc[ii] = new TreeArcs;
     }

   if (! Treearcfile)
     {
      MessageBox ("Unable to open file: Tree arc file", "FileError",
                   MB_OK | MB_ICONEXCLAMATION);
     }
   else
     {
      // reads the file header
      Treearcfile >> charSnode >> charEnode;
      narct = 0;
        do
         {
          narct ++;
          Treearcfile >> treearc[narct] -> Snode
                      >> treearc[narct] -> Enode;

         } while (! Treearcfile.eof());
```

```
    //if (ARCfile.eof() == true)
       //MessageBox ("Finished read the ARC points", "ARC file complete",
                  //MB_OK | MB_ICONINFORMATION);
    }
   Treearcfile.close();
  }

/*void TView3D :: ReadSegment()
  {
   string charX, charY, charHreq, charSegNr;

   // to handle the contour segments


   //ifstream Segmentfile("c:\\data\\pent.con");
   //ifstream Segmentfile("c:\\data\\lochgrd4.con");
   //ifstream Segmentfile("c:\\data\\sercom30.con");
   //ifstream Segmentfile("c:\\data\\sercom50.con");
   //ifstream Segmentfile("c:\\data\\sercom10.con");
   ifstream SegmentFile = "c:\\data\\lochgrd.con";

   // Allocate memory for tree arcs
   for (int sg = 0; sg < maxsegment; sg ++)
     {
      Seg[sg] = new Segment;
     }

   if (! SegmentFile)
     {
      MessageBox ("Unable to open file: Segment contours file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);
     }
   else
     {
      // reads the header for the segment file
      SegmentFile >> charX >> charY >> charHreq >> charSegNr;
      prevSegNr = -999;
      SegmentFile >> x >> y >> h >> SegNr;
      startx = x;
      starty = y;

      int nseg = 0;
        do
         {
          nseg ++;
          SegmentFile >> Seg[nseg]->x
                      >> Seg[nseg]->y
                      >> Seg[nseg]->Hreq
                      >> Seg[nseg]->SegNr;

         } while (! SegmentFile.eof());

    //if (ARCfile.eof() == true)
       //MessageBox ("Finished read the ARC points", "ARC file complete",
                  //MB_OK | MB_ICONINFORMATION);
    }
   SegmentFile.close();
  }
*/

void TView3D :: Transform(int MaxY, float& x, float& y)
```

```
  {
   x = (x - xmin) * scaleX;
   y = MaxY - ((y - ymin) * scaleY);
  }

double TView3D :: DotProduct(Vector v1, Vector v2)
  {
   double dp;
   dp = (v1.x * v2.x) + (v1.y * v2.y) + (v1.z * v2.z);
   return dp;
  }

double TView3D :: Magnitude(Vector v)
  {
   double Mag;
   Mag = sqrt((v.x * v.x) + (v.y * v.y) + (v.z * v.z));
   return Mag;
  }

void TView3D :: Normalize(Vector v1, Vector& v2)
  {
   double denominator;

   denominator = Magnitude(v1);
   if (denominator == 0)
      v2 = v1;
   else
     {
      v2.x = v1.x / denominator;
      v2.y = v1.y / denominator;
      v2.z = v1.z / denominator;
     }
  }

void TView3D :: VectorMatrix(Vector v1, Matrix T, Vector& v2)
  {
   T[3][2] = rho;
   T[3][3] = 1;
   v2.x = v1.x * T[0][0] + v1.y * T[1][0] + v1.z * T[2][0] + T[3][0];
   v2.y = v1.x * T[0][1] + v1.y * T[1][1] + v1.z * T[2][1] + T[3][1];
   v2.z = v1.x * T[0][2] + v1.y * T[1][2] + v1.z * T[2][2] + T[3][2];
  }

void TView3D :: SetZeroMatrix(Matrix& m)
  {
   for (int i = 0; i < 3; i ++)
     for (int j = 0; j < 3; j ++)
       {
        m[i][j] = 0;
       }
  }

void TView3D :: SetIdentityMatrix(Matrix& m)
  {
   SetZeroMatrix(m);
   for (int diag = 0; diag < 3; diag ++)
     for (diag = 0; diag < 3; diag ++)
       {
        m[diag][diag] = 1;
       }
  }
```

```
double TView3D :: InRadians(double degrees)
  {
   return degrees * M_PI / 180;
  }

void TView3D :: SetViewVariables(double& x_view, double& y_view, double&
z_view)
  {
   sin_theta = sin((InRadians(theta)));
   cos_theta = cos((InRadians(theta)));
   sin_phi   = sin((InRadians(phi)));
   cos_phi   = cos((InRadians(phi)));
   x_view = rho * cos_theta * sin_phi;
   y_view = rho * sin_theta * sin_phi;
   z_view = rho * cos_phi;
  }

void TView3D :: SetViewTransformationMatrix(Matrix viewT)
  {
   viewT[0][0] = - sin_theta;
   viewT[0][1] = - cos_theta * cos_phi;
   viewT[0][2] = - cos_theta * sin_phi;
   viewT[1][0] =   cos_theta;
   viewT[1][1] = - sin_theta * cos_phi;
   viewT[1][2] = - sin_theta * sin_phi;
   viewT[2][1] =   sin_phi;
   viewT[2][2] = - cos_phi;
   viewT[3][2] =   rho;
   viewT[3][3] = 1;
  }

void TView3D :: Perspective(double xx, double yy, double zz,
                            double& screen_x, double& screen_y)
  {
   double temp_x, temp_y;

   xmid = (MaxX + 1) / 2;
   ymid = (MaxY + 1) / 2;
   temp_x = ViewPlaneDist * (xx / zz);
   temp_y = ViewPlaneDist * (yy / zz);
   screen_x = xmid + temp_x;
   screen_y = ymid + temp_y;
  }

void TView3D :: GetMinMax(double& xmax, double& ymax, double& zmax,
                          double& xmin, double& ymin, double& zmin,
                          double& xlength, double& ylength, double& zlength)
  {
   //ifstream XYZfile = "c:\\data\\test.xyz";
   ifstream XYZfile = "c:\\data\\lochass.xyz";

   if (! XYZfile)
      MessageBox ("Unable to open file: XYZ file", "FileError",
                  MB_OK | MB_ICONEXCLAMATION);

   string charX, charY, charZ;
   double xa, ya, za;
   // reads the file header
   XYZfile >> charX >> charY >> charZ;

   XYZfile >> xa >> ya >> za;
```

```
      xmin = xa;
      xmax = xa;

      ymin = ya;
      ymax = ya;

      zmin = za;
      zmax = za;

      do
        {
         XYZfile >> xa >> ya >> za;

         if (xmin > xa)
             xmin = xa;

         if (ymin > ya)
             ymin = ya;

         if (zmin > za)
             zmin = za;

         if (xmax < xa)
             xmax = xa;

         if (ymax < ya)
             ymax = ya;

         if (zmax < za)
             zmax = za;
         } while (! XYZfile.eof());

      xlength = xmax - xmin;
      ylength = ymax - ymin;
      zlength = zmax - zmin;
  }

void TView3D :: Paint(TDC& dc, bool, TRect&)
  {
   GetMinMax(xmax, ymax, zmax, xmin, ymin, zmin,
             xlength, ylength, zlength);

   ReadXYZ();
   ReadXYZ2();
   ReadTIN();
   //ReadTIN2();
   ReadARC();       // reads the ARC
   ReadRoofArc();   // the roof file
   ReadTreeXYZ();   // the tree points
   ReadTreeArc();   // the tree arcs
   //ReadSegment();  // the contour segments

   // to set the backgriund color
   static const TColor color (RGB(0, 0, 0));  // RGB - black
   TWindow :: SetBkgndColor(color);

   TRect rect;
   :: GetClientRect(HWindow, &rect);

   // set the origin in the middle of the window client area
   dc.SetMapMode(MM_ANISOTROPIC);
```

```
/*dc.SetWindowExt(TSize((xmax-xmin), (ymax-ymin)));
dc.SetWindowOrg(TPoint(xmin, ymin));
dc.SetViewportExt(GetClientRect().Size());
dc.SetViewportOrg(TPoint(0, 0));
*/

SetViewportOrgEx(dc, rect.left-180, rect.top-300, &POINT());
// the above parameter is for the test data

// SetViewportOrgEx(dc, rect.left, rect.top, &POINT());
// the above parameter for the trylake.xyz data

//SetViewportOrgEx(dc, rect.right/2, rect.top+20, &POINT());

scaleX = (rect.right - rect.left) / xlength;
scaleY = (rect.bottom - rect.top) / ylength;

SetViewVariables(x_view, y_view, z_view);
SetViewTransformationMatrix(viewT);

// Select pen style for TENs points
TPen pen3 = TPen(RGB(250, 0, 0), 3, PS_SOLID);
SelectObject(dc, pen3);

for (k = 1; k < npnt; k ++)
  {
   VecWorld.x = pnt[k]->x;
   VecWorld.y = pnt[k]->y;
   VecWorld.z = pnt[k]->z;

   VectorMatrix(VecWorld, viewT, VecView);

   pnt[k]->x = VecView.x;
   pnt[k]->y = VecView.y;
   pnt[k]->z = VecView.z;

   pnt[k]->x = (pnt[k]->x - xmin)* scaleX;
   pnt[k]->y = (rect.bottom - rect.top) - ((pnt[k]->y - ymin) * scaleY);

   Perspective(pnt[k]->x, pnt[k]->y, pnt[k]->z, screen_x, screen_y);

   pnt[k]->x = screen_x;
   pnt[k]->y = screen_y;

   // to draw TENs points
   dc.MoveTo(pnt[k]->x, pnt[k]->y);
   dc.LineTo(pnt[k]->x, pnt[k]->y);

  }
SelectObject(dc, pen3);
DeleteObject(pen3);


// to handle points for TINS
// Select pen style for points
/*TPen pentinpnt = TPen(RGB(10, 10, 10), 3, PS_SOLID);
SelectObject(dc, pentinpnt);
for (int kk = 1; kk < npntt; kk ++)
  {
   VecWorld.x = pntt[kk]->x;
   VecWorld.y = pntt[kk]->y;
```

```
      VecWorld.z = pntt[kk]->z;

      VectorMatrix(VecWorld, viewT, VecView);

      pntt[kk]->x = VecView.x;
      pntt[kk]->y = VecView.y;
      pntt[kk]->z = VecView.z;

      pntt[kk]->x = (pntt[kk]->x - xmin)* scaleX;
        pntt[kk]->y = (rect.bottom - rect.top) - ((pntt[kk]->y - ymin) *
scaleY);

     Perspective(pntt[kk]->x, pntt[kk]->y, pntt[kk]->z, screen_x, screen_y);

      pntt[kk]->x = screen_x;
      pntt[kk]->y = screen_y;

      // to draw points
      dc.MoveTo(pntt[kk]->x, pntt[kk]->y);
      dc.LineTo(pntt[kk]->x, pntt[kk]->y);

    }
   SelectObject(dc, pentinpnt);
   DeleteObject(pentinpnt);
   */

   // set pen style for polygon (i.e. constrained edges) display
   TPen penpoly = TPen(RGB(255, 0, 0), 2, PS_SOLID);
   SelectObject(dc, penpoly);

   /*for (int r = 1; r < narc; r ++)
     {
      dc.MoveTo(pntt[poly_ptr[r]->Snode]->x,
                pntt[poly_ptr[r]->Snode]->y);
      dc.LineTo(pntt[poly_ptr[r]->Enode]->x,
                pntt[poly_ptr[r]->Enode]->y);
      dc.LineTo(pntt[poly_ptr[r]->Snode]->x,
                pntt[poly_ptr[r]->Snode]->y);
     }
    */
   for (int r = 1; r < narc; r ++)
     {
      dc.MoveTo(pnt[poly_ptr[r]->Snode]->x,
                pnt[poly_ptr[r]->Snode]->y);
      dc.LineTo(pnt[poly_ptr[r]->Enode]->x,
                pnt[poly_ptr[r]->Enode]->y);
      dc.LineTo(pnt[poly_ptr[r]->Snode]->x,
                pnt[poly_ptr[r]->Snode]->y);
     }

   SelectObject(dc, penpoly);
   DeleteObject(penpoly);


   // Select pen style (different colour)
   TPen pen1 = TPen(RGB(255, 255, 255), 1, PS_SOLID);
   SelectObject(dc, pen1);

   /*
   // Display the 8 corners of the 3D space
   dc.MoveTo(pnt[1]->x, pnt[1]->y);
```

```
dc.LineTo(pnt[2]->x, pnt[2]->y);
dc.LineTo(pnt[3]->x, pnt[3]->y);
dc.LineTo(pnt[4]->x, pnt[4]->y);
dc.LineTo(pnt[1]->x, pnt[1]->y);
dc.LineTo(pnt[5]->x, pnt[5]->y);
dc.LineTo(pnt[6]->x, pnt[6]->y);
dc.LineTo(pnt[7]->x, pnt[7]->y);
dc.LineTo(pnt[8]->x, pnt[8]->y);
dc.LineTo(pnt[5]->x, pnt[5]->y);
dc.MoveTo(pnt[3]->x, pnt[3]->y);
dc.LineTo(pnt[7]->x, pnt[7]->y);
dc.MoveTo(pnt[2]->x, pnt[2]->y);
dc.LineTo(pnt[6]->x, pnt[6]->y);
dc.MoveTo(pnt[4]->x, pnt[4]->y);
dc.LineTo(pnt[8]->x, pnt[8]->y);

SelectObject(dc, pen1);
DeleteObject(pen1);
*/

// to handle TEN
// Another pen style (different colour)
TPen penten = TPen(RGB(0, 255, 0), 1, PS_SOLID);
SelectObject(dc, penten);

for (k = 1; k < ntin; k ++)
  {
   dc.MoveTo(pnt[tin[k]->Node1]->x,
             pnt[tin[k]->Node1]->y);
   dc.LineTo(pnt[tin[k]->Node2]->x,
             pnt[tin[k]->Node2]->y);
   dc.LineTo(pnt[tin[k]->Node3]->x,
             pnt[tin[k]->Node3]->y);
   dc.LineTo(pnt[tin[k]->Node1]->x,
             pnt[tin[k]->Node1]->y);
  }

SelectObject(dc, penten);
DeleteObject(penten);

/*
// to handle TIN
TPen pentintop = TPen(RGB(255, 0, 0), 1, PS_SOLID);
SelectObject(dc, pentintop);

for (k = 1; k < ntinn; k ++)
  {
   dc.MoveTo(pntt[tinn[k]->Node1]->x,
             pntt[tinn[k]->Node1]->y);
   dc.LineTo(pntt[tinn[k]->Node2]->x,
             pntt[tinn[k]->Node2]->y);
   dc.LineTo(pntt[tinn[k]->Node3]->x,
             pntt[tinn[k]->Node3]->y);
   dc.LineTo(pntt[tinn[k]->Node1]->x,
             pntt[tinn[k]->Node1]->y);
  }

SelectObject(dc, pentintop);
DeleteObject(pentintop);
*/
```

```
/*
// to handle boreholes
TPen penbore = TPen(RGB(255, 255, 10), 1, PS_SOLID);
SelectObject(dc, penbore);

// this is for lakebore data
dc.MoveTo(pnt[9]->x, pnt[9]->y);
dc.LineTo(pnt[10]->x, pnt[10]->y);
dc.LineTo(pnt[11]->x, pnt[11]->y);
dc.LineTo(pnt[12]->x, pnt[12]->y);
dc.LineTo(pnt[13]->x, pnt[13]->y);
dc.LineTo(pnt[14]->x, pnt[14]->y);

dc.MoveTo(pnt[15]->x, pnt[15]->y);
dc.LineTo(pnt[16]->x, pnt[16]->y);
dc.LineTo(pnt[17]->x, pnt[17]->y);
dc.LineTo(pnt[18]->x, pnt[18]->y);
dc.LineTo(pnt[19]->x, pnt[19]->y);
dc.LineTo(pnt[20]->x, pnt[20]->y);

dc.MoveTo(pnt[21]->x, pnt[21]->y);
dc.LineTo(pnt[22]->x, pnt[22]->y);
dc.LineTo(pnt[23]->x, pnt[23]->y);
dc.LineTo(pnt[24]->x, pnt[24]->y);
dc.LineTo(pnt[25]->x, pnt[25]->y);
dc.LineTo(pnt[26]->x, pnt[26]->y);

dc.MoveTo(pnt[27]->x, pnt[27]->y);
dc.LineTo(pnt[28]->x, pnt[28]->y);
dc.LineTo(pnt[29]->x, pnt[29]->y);
dc.LineTo(pnt[30]->x, pnt[30]->y);
dc.LineTo(pnt[31]->x, pnt[31]->y);
dc.LineTo(pnt[32]->x, pnt[32]->y);

dc.MoveTo(pnt[33]->x, pnt[33]->y);
dc.LineTo(pnt[34]->x, pnt[34]->y);
dc.LineTo(pnt[35]->x, pnt[35]->y);
dc.LineTo(pnt[36]->x, pnt[36]->y);
dc.LineTo(pnt[37]->x, pnt[37]->y);
dc.LineTo(pnt[38]->x, pnt[38]->y);

dc.MoveTo(pnt[39]->x, pnt[39]->y);
dc.LineTo(pnt[40]->x, pnt[40]->y);
dc.LineTo(pnt[41]->x, pnt[41]->y);
dc.LineTo(pnt[42]->x, pnt[42]->y);
dc.LineTo(pnt[43]->x, pnt[43]->y);
dc.LineTo(pnt[44]->x, pnt[44]->y);

SelectObject(dc, penbore);
DeleteObject(penbore);
*/

// to handle buildings
// Select pen style for building roof points
TPen penbuildpnt = TPen(RGB(250, 0, 0), 3, PS_SOLID);
SelectObject(dc, penbuildpnt);

for (int k = 1; k < npntt; k ++)
  {
   VecWorld.x = pntt[k]->x;
   VecWorld.y = pntt[k]->y;
```

```
    VecWorld.z = pntt[k]->z;

    VectorMatrix(VecWorld, viewT, VecView);

    pntt[k]->x = VecView.x;
    pntt[k]->y = VecView.y;
    pntt[k]->z = VecView.z;

    pntt[k]->x = (pntt[k]->x - xmin)* scaleX;
   pntt[k]->y = (rect.bottom - rect.top) - ((pntt[k]->y - ymin) * scaleY);

    Perspective(pntt[k]->x, pntt[k]->y, pntt[k]->z, screen_x, screen_y);

    pntt[k]->x = screen_x;
    pntt[k]->y = screen_y;

    // to draw building roof points
    dc.MoveTo(pntt[k]->x, pntt[k]->y);
    dc.LineTo(pntt[k]->x, pntt[k]->y);
   }
SelectObject(dc, penbuildpnt);
DeleteObject(penbuildpnt);

// set pen style for the building roof lines
TPen penbuildline = TPen(RGB(0, 255, 255), 2, PS_SOLID);
SelectObject(dc, penbuildline);

for (int f = 1; f < narcr; f ++)
  {
   dc.MoveTo(pntt[roofarc[f]->Snode]->x,
             pntt[roofarc[f]->Snode]->y);
   dc.LineTo(pntt[roofarc[f]->Enode]->x,
             pntt[roofarc[f]->Enode]->y);
   dc.LineTo(pntt[roofarc[f]->Snode]->x,
             pntt[roofarc[f]->Snode]->y);
  }

SelectObject(dc, penbuildline);
DeleteObject(penbuildline);


// to handle trees
// Select pen style for trees points
TPen pentreepnt = TPen(RGB(250, 0, 0), 2, PS_SOLID);
SelectObject(dc, pentreepnt);

for (int kt = 1; kt < ntree; kt ++)
  {
   VecWorld.x = treepnts[kt]->x;
   VecWorld.y = treepnts[kt]->y;
   VecWorld.z = treepnts[kt]->z;

   VectorMatrix(VecWorld, viewT, VecView);

   treepnts[kt]->x = VecView.x;
   treepnts[kt]->y = VecView.y;
   treepnts[kt]->z = VecView.z;

   treepnts[kt]->x = (treepnts[kt]->x - xmin)* scaleX;
   treepnts[kt]->y = (rect.bottom - rect.top) - ((treepnts[kt]->y - ymin)
                                                 * scaleY);
```

```
    Perspective(treepnts[kt]->x, treepnts[kt]->y, treepnts[kt]->z, screen_x,
screen_y);

      treepnts[kt]->x = screen_x;
      treepnts[kt]->y = screen_y;

      // to draw building roof points
      dc.MoveTo(treepnts[kt]->x, treepnts[kt]->y);
      dc.LineTo(treepnts[kt]->x, treepnts[kt]->y);
     }
   SelectObject(dc, pentreepnt);
   DeleteObject(pentreepnt);

   // set pen style for the tree arcs
   TPen pentreearcs = TPen(RGB(255, 0, 0), 2, PS_SOLID);
   SelectObject(dc, pentreearcs);

   for (int ft = 1; ft < narct; ft ++)
     {
      dc.MoveTo(treepnts[treearc[ft]->Snode]->x,
                treepnts[treearc[ft]->Snode]->y);
      dc.LineTo(treepnts[treearc[ft]->Enode]->x,
                treepnts[treearc[ft]->Enode]->y);
      dc.LineTo(treepnts[treearc[ft]->Snode]->x,
                treepnts[treearc[ft]->Snode]->y);
     }

   SelectObject(dc, pentreearcs);
   DeleteObject(pentreearcs);

   // to handle the contour segments
   // set pen style for the segments

   // colour guides:
   // Light Magenta: (255 0 255)
   // Light Yellow : (255, 255, 0)
   // Light Grey   : (192, 192, 192)
   // Grey         : (128, 128, 128)
   // Light Cyan   : (0, 255, 255)

   TPen pensegment = TPen(RGB(255, 255, 0), 2, PS_SOLID);
   SelectObject(dc, pensegment);

   // to handle the contour segments

   //ifstream SegmentFile("c:\\data\\pent.con");
   ifstream SegmentFile("c:\\data\\lochgrd4.con");
   //ifstream SegmentFile("c:\\data\\sercom30.con");
   //ifstream SegmentFile("c:\\data\\sercom50.con");
   //ifstream SegmentFile("c:\\data\\sercom10.con");
   //ifstream SegmentFile = "c:\\data\\lochgrd.con";

   string charX, charY, charHreq, charSegNr;

   // Allocate memory for contour segments
   for (int sg = 0; sg < maxsegment; sg ++)
     {
      Seg[sg] = new Segment;
     }
```

```
if (! SegmentFile)
  {
   MessageBox ("Unable to open file: Segment contours file", "FileError",
               MB_OK | MB_ICONEXCLAMATION);
  }
else
  {
   // reads the header for the segment file
   SegmentFile >> charX >> charY >> charHreq >> charSegNr;
   prevSegNr = -999;
   SegmentFile >> x >> y >> h >> SegNr;
   Transform(MaxY, x, y);
   VecWorld.x = x;
   VecWorld.y = y;
   VecWorld.z = h;

   VectorMatrix(VecWorld, viewT, VecView);

   x = VecView.x;
   y = VecView.y;
   h = VecView.z;

   x = (x - xmin)* scaleX;
   y = (rect.bottom - rect.top) - ((y - ymin) * scaleY);

   Perspective(x, y, h, screen_x, screen_y);

   x = screen_x;
   y = screen_y;

   startx = x;
   starty = y;

   while (! SegmentFile.eof())
     {
      SegmentFile >> nextx >> nexty >> nexth >> nextSegNr;
      Transform(MaxY, nextx, nexty);
      VecWorld.x = nextx;
      VecWorld.y = nexty;
      VecWorld.z = nexth;

      VectorMatrix(VecWorld, viewT, VecView);

      nextx = VecView.x;
      nexty = VecView.y;
      nexth = VecView.z;

      nextx = (nextx - xmin)* scaleX;
      nexty = (rect.bottom - rect.top) - ((nexty - ymin) * scaleY);

      Perspective(nextx, nexty, nexth, screen_x, screen_y);

      nextx = screen_x;
      nexty = screen_y;

      if (SegNr != prevSegNr)
        {
         dc.MoveTo(x, y);
         startx = x;
         starty = y;
        }
```

```
          else
            {
             if ( (x == startx) && (y == starty) )
               {
                if (SegNr == nextSegNr)
                   dc.MoveTo(x, y);
                else
                   dc.LineTo(x, y);
               }
             else
             dc.LineTo(x, y);
            }

          prevSegNr = SegNr;
          SegNr = nextSegNr;

          x = nextx;
          y = nexty;
          h = nexth;
         }

       dc.MoveTo(x, y);
       SegmentFile.close();
      }

      SelectObject(dc, pensegment);
      DeleteObject(pensegment);

    DeallocateMemory();
   }

void TView3D :: CmInputFilesItem1()
   {
    ReadXYZ();
   }

void TView3D :: CmPopupItem1()
   {
    ReadTIN();
   }

// Force the window to repaint if resize
void TView3D :: EvSize(uint, TSize&)
   {
    :: InvalidateRect(HWindow, 0, true);
   }

void TView3D :: CmClear()
   {
    // to clear and redraw
    Invalidate();
   }


void TView3D :: CmAbout()
   {
    MessageBox ("3D TINs display with menu - by Alias Abdul-Rahman, 1999",
                "About the 2D/3D TIN", MB_OK | MB_ICONINFORMATION);
   }
```

```
void TView3DWindowApp :: InitMainWindow()
  {
   // construct the decorated MDI frame window
   TDecoratedFrame* frame = new TDecoratedFrame(0,
                     "3D Viewing (wire-frame)", new TView3D);

   // construct the status bar
   TStatusBar* sb = new TStatusBar(frame, TGadget :: Recessed,
                                    TStatusBar :: CapsLock | TStatusBar ::
NumLock);

   // construct the control bar
   TControlBar* cb = new TControlBar(frame);

   cb -> Insert(*new TButtonGadget(CM_INPUTFILESITEM1, CM_INPUTFILESITEM1,
                     TButtonGadget :: Command));
   cb -> Insert(*new TButtonGadget(CM_POPUPITEM1, CM_POPUPITEM1,
                     TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_CLEAR, CM_CLEAR,
                     TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_FILEEXIT, CM_FILEEXIT,
                     TButtonGadget :: Command));

   cb -> Insert(*new TSeparatorGadget);

   cb -> Insert(*new TButtonGadget(CM_ABOUT, CM_ABOUT,
                     TButtonGadget :: Command));

   cb -> SetHintMode(TGadgetWindow :: EnterHints);

   // set client area to the application workspace
   frame -> SetBkgndColor(::GetSysColor(COLOR_APPWORKSPACE));

   // insert the status and control bar into the frame
   frame -> Insert(*sb, TDecoratedFrame :: Bottom);
   frame -> Insert(*cb, TDecoratedFrame :: Top);

   // set the main window and its menu
   SetMainWindow(frame);
   GetMainWindow() -> AssignMenu("COMMANDS");
   EnableCtl3d(true);
  }


int OwlMain(int /*argc*/, char* /*argv*/ [])
  {
   return TView3DWindowApp().Run();
  }
```

```
//************************************************************************
// Project: view3d
// File: view3d.rh
//
//************************************************************************

#define CM_CLEAR           1125
#define CM_FILEEXIT        1126
#define CM_ABOUT           1127
#define CM_INPUTFILESITEM1    18874 // for XYZ file
#define CM_POPUPITEM1      18873 // for TIN file
```

```
//**************************************************************************
// Project: view3D
// File: view3D.rc
//
//**************************************************************************


#include "view3d.rh"

COMMANDS MENU
{
 POPUP "&InputFiles"
 {
  MENUITEM "&XYZ File", CM_INPUTFILESITEM1
  MENUITEM "&TIN File", CM_POPUPITEM1
 }

 POPUP "&Redraw"
 {
  MENUITEM "&Clear and Redraw", CM_CLEAR
  MENUITEM "E&xit", CM_FILEEXIT
 }

 MENUITEM "&About", CM_ABOUT
}

STRINGTABLE
  {
   CM_INPUTFILESITEM1,   "Get the XYZ file"
   CM_POPUPITEM1,        "Get the TIN file"
   CM_CLEAR,             "Redraw the TINs"
   CM_FILEEXIT,          "Exit the program ..."
   CM_ABOUT,             "About the program ..."
  }

CM_CLEAR BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 91 77 77 77 77 77 77 77 77 99'
 '00 00 79 77 77 77 77 77 77 77 79 17 00 00 77 17'
 '77 87 77 77 77 79 91 77 00 00 77 77 80 08 88 87'
 '77 77 77 77 00 00 77 78 0A 00 08 88 77 77 77 77'
 '00 00 77 78 0A 0A A0 08 88 87 77 77 00 00 77 78'
 '0A 0A AA A0 88 88 77 77 00 00 77 80 AA AA AA AA'
 '00 88 77 77 00 00 77 80 AA AA AA A0 A0 88 77 77'
 '00 00 77 0A AA A0 AA 0A A0 88 71 17 00 00 77 0A'
 'AA A0 00 AA A0 88 79 97 00 00 78 0A AA A0 AA AA'
 'A0 88 77 77 00 00 78 AA A0 0A AA AA A0 88 77 77'
 '00 00 78 88 87 78 0A AA A0 88 77 77 00 00 77 77'
 '77 77 80 AA A0 88 77 77 00 00 77 71 77 77 78 0A'
 'A0 88 77 77 00 00 77 19 77 77 77 80 A0 88 77 77'
 '00 00 77 97 77 77 77 78 A0 88 79 77 00 00 71 77'
 '77 77 77 77 88 87 71 19 00 00 19 77 77 77 77 77'
 '77 77 77 71 00 00'
```

```
}


CM_FILEEXIT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 EE EE EE EE EE EE EE EE EE EE'
 '00 00 EE EE FF EF EF FE EE EE EE EE 00 00 EE EE'
 'FF EE FE EF EE EE EE EE 00 00 EE EE FF EF FF FF'
 'EE EE EE EE 00 00 EE EE EF EE EF FF EE EE EE EE'
 '00 00 88 88 88 88 88 88 88 88 88 88 00 00 66 66'
 '8A AA A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A A9 A8 77 86 66 66 66 00 00 66 66'
 '8A A9 A8 77 86 66 66 66 00 00 66 66 8A AA A8 77'
 '86 66 66 66 00 00 66 66 8A AA A8 77 86 66 66 66'
 '00 00 66 66 8A AA A8 77 86 66 66 66 00 00 66 66'
 '8A A8 87 77 86 66 66 66 00 00 66 66 8A 87 77 77'
 '86 66 66 66 00 00 66 66 88 77 77 77 86 66 66 66'
 '00 00 66 66 88 88 88 88 86 66 66 66 00 00 66 66'
 '66 66 66 66 66 66 66 66 00 00 66 66 66 66 66 66'
 '66 66 66 66 00 00'
}


CM_ABOUT BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB B8 88 88 BB BB BB BB'
 '00 00 BB BB BB B8 00 88 BB BB BB BB 00 00 BB BB'
 'BB B8 00 BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00 BB BB BB B8 00 88 BB BB BB BB'
 '00 00 BB BB BB BB 80 08 BB BB BB BB 00 00 BB BB'
 'B8 8B B8 08 8B BB BB BB 00 00 BB BB B8 80 BB 00'
 'BB BB BB BB 00 00 BB BB B8 80 BB 00 8B BB BB BB'
 '00 00 BB BB BB 88 88 88 BB BB BB BB 00 00 BB BB'
 'BB B8 88 88 8B BB BB BB 00 00 BB BB BB B8 88 88'
 'BB BB BB BB 00 00 BB BB BB BB BB BB BB BB BB BB'
 '00 00 BB BB BB BB BB BB BB BB BB BB 00 00 BB BB'
 'BB BB BB BB BB BB BB BB 00 00 BB BB BB BB BB BB'
 'BB BB BB BB 00 00'
}

CM_INPUTFILESITEM1 BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
```

```
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 77 77 77 77 77 77 77 77 77 77'
 '00 00 77 00 00 00 00 00 00 00 08 77 00 00 77 0A'
 'AA AA AA AA AA AA 08 77 00 00 77 0A 99 9A AA AA'
 'AA AA 08 77 00 00 77 0A AA AA AA AA AA AA 08 77'
 '00 00 77 0A AA AA AA AA AA AA 08 77 00 00 77 00'
 '00 00 00 00 00 00 08 77 00 00 77 88 88 88 88 88'
 '88 88 88 77 00 00 77 77 77 77 77 77 77 77 77 77'
 '00 00 77 00 00 00 00 00 00 77 77 77 00 00 77 0F'
 'FF FF FF FF F0 77 77 77 00 00 77 0F 00 0F FF FF'
 'F0 77 77 77 00 00 77 0F FF FF F0 F0 F0 77 77 77'
 '00 00 77 0F 0F 0F 0F 0F F0 77 77 77 00 00 77 0F'
 'FF FF FF FF 08 77 77 77 00 00 77 0F 00 0F 0F FF'
 '88 77 77 77 00 00 77 0F FF FF FF 00 88 77 77 77'
 '00 00 77 00 00 00 00 88 88 77 77 77 00 00 77 77'
 '77 77 77 77 77 77 77 77 00 00 77 77 77 77 77 77'
 '77 77 77 77 00 00'
}


CM_POPUPITEM1 BITMAP
{
 '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 14 00 00 00 14 00 00 00 01 00 04 00 00 00'
 '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 77 77 77 77 77 77 77 77 77 77'
 '00 00 77 00 00 00 00 00 00 00 08 77 00 00 77 0A'
 'AA AA AA AA AA AA 08 77 00 00 77 0A 99 9A AA AA'
 'AA AA 08 77 00 00 77 0A AA AA AA AA AA AA 08 77'
 '00 00 77 0A AA AA AA AA AA AA 08 77 00 00 77 00'
 '00 00 00 00 00 00 08 77 00 00 77 88 88 88 88 88'
 '88 88 88 77 00 00 77 77 77 77 77 77 77 77 77 77'
 '00 00 77 70 00 00 00 00 00 07 77 77 00 00 77 70'
 'FF FF FF FF FF 07 77 77 00 00 77 70 F0 FF FF 00'
 'FF 07 77 77 00 00 77 70 FF FF 00 FF FF 07 77 77'
 '00 00 77 70 F0 0F FF FF FF 07 77 77 00 00 77 70'
 'FF FF F0 0F FF 07 77 77 00 00 77 70 FF FF FF FF'
 'F0 87 77 77 00 00 77 70 FF 00 0F FF 08 87 77 77'
 '00 00 77 70 FF FF FF F0 88 87 77 77 00 00 77 70'
 '00 00 00 88 88 87 77 77 00 00 77 77 77 77 77 77'
 '77 77 77 77 00 00'
}
```

POET Database Compatible Code

```
//*************************************************************//
//  Project: TIN-based Spatial Modelling using POET            //
//  File: main.cpp                                             //
//  Author: Alias Abdul-Rahman (c) Feb., 2000                  //
//                                                             //
//*************************************************************//

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <fstream.h>
#include <wtypes.h>
#include <conio.h>

// POET include file
#include <poet.hxx>

// PTXX generated file
#include "tnode.cxx"
#include "tedge.cxx"
#include "tpoly.cxx"
#include "tsolid.cxx"


int main(int argc, char *argv[])
  {
   // Initiliase the POET database
   InitPOET();
   PtBase* pObjBase;
   int err = PtBase :: POET()->GetBase("LOCAL", "base", pObjBase);
   if (err < 0)
     {
      PtBase :: POET()->UngetBase(pObjBase);
      DeinitPOET();
      return err;
     }

   // to handle the TNode class
   TNode Node;
   Node.GetXYZCoordinates();
   Node.GetBoreholeCoordinates();
   Node.Get2Node();
   Node.NodeAttribute();

   // to store the Nodes objects in the database
   Node.Assign(pObjBase);
   err = Node.Store();

   // to handle the TEdge class
   TEdge Edge;
   Edge.ReadARCs();
   Edge.GetOneArc();
```

```
  Edge.EdgeAttribute();
  Edge.GetArcLength();

  // to stote the Edge objects in the database
  Edge.Assign(pObjBase);
  err = Edge.Store();

  // to handle the TPoly class
  TPoly Poly;
  Poly.ReadTINs();
  Poly.GetTINNeighbour();
  Poly.GetPolyArea();

  // to stote the TPoly object in the database
  Poly.Assign(pObjBase);
  err = Poly.Store();

  // to handle the TSolid class
  TSolid Solid;
  Solid.ReadTENs();
  Solid.GetVolume();

  // to store the Solid objects in the database
  Solid.Assign(pObjBase);
  err = Solid.Store();

  // to close the database
  PtBase :: POET()->UngetBase(pObjBase);
  DeinitPOET();
}
```

```
//*************************************************************//
//  Project: TIN-based Spatial Modelling using POET            //
//  File: tnode.cpp                                            //
//  Author: Alias Abdul-Rahman (c) Feb., 2000                  //
//                                                             //
//*************************************************************//

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <fstream.h>
#include <string>
#include <conio.h>


// POET compiler includes
#include <poet.hxx>

// PTXX generated files
#include "tnode.hxx"

#include "xyzcontainer.hxx"
#include "nodenamecontainer.hxx"

// Borland C++ includes
//#include "tnode.h"
//#include "xyzcontainer.h"
//#include "nodenamecontainer.h"


#define maxpoint 10000
#define maxnodename 100

// forward declaration
XYZContainer Point[maxpoint];
NodeNameContainer NodeAtr[maxnodename];
//NodeAtrContainer NodeAtr[maxnodename];

// implementation of class TNode
// the constructor
TNode :: TNode()
  {

  }

// the destructor
TNode :: ~TNode()
  {

  }


void TNode :: PressAnyKey()
  {
   cout << endl;
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }
```

```
void TNode :: GetXYZCoordinates()
  {
   string charX, charY, charZ;

   // to read the XYZ file
   ifstream XYZfile("c:\\data\\lochass.xyz");
   if (! XYZfile)
     cout << "Could not open the file" << endl;
   else
     {
      // reads the file header
      XYZfile >> charX >> charY >> charZ;
      int npnt = 0;
      do
        {
         npnt ++;
         printf("\r");
         cout << "reading points ...." << npnt;
         XYZfile >> Point[npnt].x >> Point[npnt].y >> Point[npnt].z;
        } while (! XYZfile.eof());
     }

   cout << endl;
   cout << endl;
   XYZfile.close();
  }

void TNode :: GetBoreholeCoordinates()
  {
   string charX, charY, charZ;

   // to read the XYZ file
   ifstream XYZBorefile("c:\\data\\lakebore.xyz");
   if (! XYZBorefile)
     cout << "Could not open the file" << endl;
   else
     {
      // reads the file header
      XYZBorefile >> charX >> charY >> charZ;
      int npnt = 0;
      do
        {
         npnt ++;
         printf("\r");
         cout << "reading points ...." << npnt;
         XYZBorefile >> Point[npnt].x >> Point[npnt].y >> Point[npnt].z;
        } while (! XYZBorefile.eof());
     }

   cout << endl;
   cout << endl;
   XYZBorefile.close();
  }


void TNode :: Get2Node()
  {
   int selectnode1, selectnode2;
   int npnt1, npnt2;
   double Node1x, Node1y;
   double Node2x, Node2y;
```

```
   cout << "The point# range: 1 to " << npnt << endl;
   cout << "Select node 1 :" << endl;
   cin >> selectnode1;
   cout << "Select node 2:" << endl;
   cin >> selectnode2;
   npnt1 = selectnode1;
   npnt2 = selectnode2;
   Node1x = Point[npnt1].x;
   Node1y = Point[npnt1].y;
   Node2x = Point[npnt2].x;
   Node2y = Point[npnt2].y;
   cout << endl;
   cout << "The selected points are:" << endl;
   cout << "    Node1  : " << npnt1 << endl;
   cout << "     (x, y): (" << Node1x << ", " << Node1y << ")" << endl;
   cout << "    Node2  : " << npnt2 << endl;
   cout << "     (x, y): (" << Node2x << ", " << Node2y << ")" << endl;
   cout << endl;
  }

void TNode :: NodeAttribute()
  {
   string charNodeNr, charNodeAtr;

   // reads the node attribute
   ifstream NodeAtrFile("c:\\data\\node.atr");

   if (! NodeAtrFile)
     cout << "Could not open the Node attribute file" << endl;
   else
     {
      // reads the file header
      NodeAtrFile >> charNodeNr >> charNodeAtr;

      int nnode = 0;
      do
        {
         nnode ++;
         printf("\r");
         cout << "reading the node attribute: " << nnode;
         NodeAtrFile >> NodeAtr[nnode].NodeNum >> NodeAtr[nnode].NodeName;
        } while ( ! NodeAtrFile.eof());
     }
   cout << endl;

  }
```

```
//*************************************************************//
//  Project: TIN-based Spatial Modelling using POET            //
//  File: tedge.cpp                                            //
//  Author: Alias Abdul-Rahman (c) Feb., 2000                  //
//                                                             //
//*************************************************************//

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <fstream.h>
#include <math.h>
#include <conio.h>

// POET compiler includes
#include <poet.hxx>

// PTXX generated files
#include "tedge.hxx"
#include "edgenamecontainer.hxx"
#include "tnode.hxx"

// Borland C++ includes
//#include "tedge.h"
//#include "edgenamecontainer.h"
//#include "arccontainer.h"


#define maxpoint 10000
#define maxarc 200
#define accy 1e-10


// forward declaration
TNode Node;
ARCContainer Arcs[maxarc];
EdgeNameContainer EdgeAtr[maxarc];
XYZContainer Point[maxpoint];

// the implementation for the TEdge class
TEdge :: TEdge()
  {

  }

TEdge :: ~TEdge()
  {

  }

void TEdge :: PressAnyKey()
  {
   cout << endl;
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

void TEdge :: ReadARCs()
```

```
  {
   string charStartNode, charEndNode;

   // reads the arc file
   ifstream ARCfile = "c:\\data\\lake.arc";

   if (!ARCfile)
     cout << "Unable to open file: ARC file" << endl;

   else
     {
      ARCfile >> charStartNode >> charEndNode;
      int narc = 0;
      do
        {
         narc ++;
         printf("\r");
         cout << "reading ARCs ...." << narc;
         ARCfile >> Arcs[narc].StartNode
                 >> Arcs[narc].EndNode;
        } while (! ARCfile.eof());
     }
   cout << endl;
   cout << endl;
   ARCfile.close();
  }


void TEdge :: GetOneArc()
  {
   int arcnum;
   int Node1, Node2;
   cout << "Enter one arc #: " << endl;
   cin >> arcnum;
   Node1 = Arcs[arcnum].StartNode;
   Node2 = Arcs[arcnum].EndNode;
   cout << "StartNode : " << Node1 << endl;
   cout << "EndNode   : " << Node2 << endl;
   //PressAnyKey();
  }

void TEdge :: GetArcLength()
  {
   int arcno;
   int ArcStartNode, ArcEndNode;
   double ArcX1, ArcY1, ArcX2, ArcY2;
   double dist;

   Node.GetXYZCoordinates();
   cout << "Enter the arc #: " << endl;
   cin >> arcno;
   ArcStartNode = Arcs[arcno].StartNode;
   ArcEndNode = Arcs[arcno].EndNode;
   cout << "StarNode: " << ArcStartNode << endl;
   cout << "EndNode : " << ArcEndNode << endl;
   ArcX1 = Node.Point[ArcStartNode].x;
   ArcY1 = Node.Point[ArcStartNode].y;
   ArcX2 = Node.Point[ArcEndNode].x;
   ArcY2 = Node.Point[ArcEndNode].y;

   dist = sqrt((ArcX2-ArcX1)*(ArcX2-ArcX1) + (ArcY2-ArcY1)*(ArcY2-ArcY1));
```

```
    cout << "Node1(x): " << ArcX1 << endl;
    cout << "Node1(y): " << ArcY1 << endl;
    cout << "The arc length: " << dist << endl;
    //PressAnyKey();
   }

int TEdge :: CheckQuadrant(float dx, float dy)
   {
    int c;

    if ((dx > 0) && (dx > 0))
      c = 1;
    if ((dx < 0) && (dy > 0))
      c = 2;
    if ((dx < 0) && (dy < 0))
      c = 3;
    if ((dx > 0) && (dy < 0))
      c = 4;
    return c;
   }

float TEdge :: Bearing(float x1, float y1, float x2, float y2)
   {
    float theta;
    float dx, dy;

    dx = x2 - x1;
    dy = y2 - y2;
    if (abs(dy) < accy)
       {
        theta = M_PI/2;
        return theta;
       }
    else
       return theta = atan(dx/dy);

   }

float TEdge :: GetArcAzimuth(float x1, float y1, float x2, float y2)
   {
    float dx, dy;
    float theta;
    int quadrant;

    dx = x2 - x1;
    dy = y2 - y1;
    if ((dx == 0) && (dy > 0))
      theta = 0;
    else if ((dx > 0) && (dy == 0))
             theta = M_PI/2;
    else if ((dx == 0) && (dy < 0))
             theta = M_PI;
    else if ((dx < 0) && (dy == 0))
             theta = 3 * M_PI/2;
    else
       {
        quadrant = CheckQuadrant(dx, dy);
        switch (quadrant)
          {
           case 1: theta = Bearing(x1, y1, x2, y2);
```

```
            break;

            case 2: theta = (2*M_PI) - abs(Bearing(x1, y1, x2, y2));
            break;

            case 3: theta = M_PI + abs(Bearing(x1, y1, x2, y2));
            break;

            case 4: theta = M_PI - abs(Bearing(x1, y1, x2, y2));
            break;
           }
       }

    return theta;
   }

void TEdge :: EdgeAttribute()
   {
    string charEdgeNr, charEdgeName;

    // reads the Edge attribute file
    ifstream EdgeAtrFile("c:\\data\\edge.atr");

    if (! EdgeAtrFile)
      cout << "Could not open the Edge attribute file" << endl;
    else
      {
       // reads the file header
       EdgeAtrFile >> charEdgeNr >> charEdgeName;

       int nedge = 0;
       do
         {
          nedge ++;
          printf("\r");
          cout << "reading the Edge attribute: " << nedge;
          EdgeAtrFile >> EdgeAtr[nedge].EdgeNum >> EdgeAtr[nedge].EdgeName;
         } while ( ! EdgeAtrFile.eof());
      }
    cout << endl;
    printf("%d   %s\n" , EdgeAtr[5].EdgeNum, EdgeAtr[5].EdgeName);
    //PressAnyKey();
   }
```

```
//*************************************************************//
//  Project: TIN-based Spatial Modelling using POET            //
//  File   : tpoly.cpp                                         //
//  Author : Alias Abdul-Rahman (c) Feb., 2000                 //
//                                                             //
//*************************************************************//

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <fstream.h>
#include <string>
#include <conio.h>

// POET includes
#include <poet.hxx>

// PTXX generated files
//#include "tpoly.hxx"

// Borland C++ include
//#include "tpoly.h"
//#include "tnode.h"


//JVG
#include "tpoly.hxx"
#include "tnode.hxx"
#include "tedge.hxx"

#define maxtriangle 10000
#define maxpoint 10000

// forward declaration
TNode Node;
TEdge Edge;

// Implementation for the class TPoly class
TPoly :: TPoly()    // constructor
  {

  }

TPoly :: ~TPoly()   // destructor
  {

  }


void TPoly :: PressAnyKey()
  {
   cout << endl;
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

void TPoly :: ReadTINs()
  {
   string charNode1, charNode2, charNode3;
```

```
   // reads the TIN file
   ifstream TINfile = "c:\\data\\lochas.tin";

   if (!TINfile)
     cout << "Unable to open file: TIN file " << endl;

   else
     {
      TINfile >> charNode1 >> charNode2 >> charNode3;
      int ntri = 0;
      do
        {
         ntri ++;
         printf("\r");
         cout << "reading TINs ..... " << ntri;
         TINfile >> Triangle[ntri].Node1
                 >> Triangle[ntri].Node2
                 >> Triangle[ntri].Node3;
        } while (! TINfile.eof());
     }

   cout << endl;
   cout << endl;
   TINfile.close();
  }

void TPoly :: GetTINNeighbour()
  {
   string charTriNum, charNumofNbr, charNbr1, charNbr2, charNbr3;

   // to read the TIN neighbour file
   ifstream TINNbrfile("c:\\data\\lochgrd.nbr");
   if (! TINNbrfile)
     cout << "Could not open the TIN neighbour file" << endl;
   else
     {
      // reads the file header
       TINNbrfile >> charTriNum >> charNumofNbr >> charNbr1 >> charNbr2 >>
charNbr3;
      int ntri = 0;
      do
        {
         ntri ++;
         printf("\r");
         cout << "reading TIN Neighbour ...." << ntri;
         TINNbrfile >> TINNbr[ntri].TriNum
                    >> TINNbr[ntri].NumofNbr
                    >> TINNbr[ntri].Nbr1
                    >> TINNbr[ntri].Nbr2
                    >> TINNbr[ntri].Nbr3;
        } while (! TINNbrfile.eof());
     }

   cout << endl;
   cout << endl;
   TINNbrfile.close();
  }

float TPoly :: GetTINArea(double x1, double y1,
                          double x2, double y2,
                          double x3, double y3)
```

```
  {
   double x12, y32;
   double x32, y12;

   float area;

   x12 = x1 - x2;
   x32 = x3 - x2;
   y12 = y1 - y2;
   y32 = y3 - y2;

   area = 0.5 * (x12*y32 - x32*y12);
   return area;
  }

void TPoly :: GetTINNodes(int t,
                          double& xNd1, double& yNd1, double& zNd1,
                          double& xNd2, double& yNd2, double& zNd2,
                          double& xNd3, double& yNd3, double& zNd3)
  {
   int Nd1, Nd2, Nd3;

   Nd1 = Triangle[t].Node1;
   Nd2 = Triangle[t].Node2;
   Nd3 = Triangle[t].Node3;

   xNd1 = Node.Point[Nd1].x;
   yNd1 = Node.Point[Nd1].y;
   zNd1 = Node.Point[Nd1].z;

   xNd2 = Node.Point[Nd2].x;
   yNd2 = Node.Point[Nd2].y;
   zNd2 = Node.Point[Nd2].z;

   xNd3 = Node.Point[Nd3].x;
   yNd3 = Node.Point[Nd3].y;
   zNd3 = Node.Point[Nd3].z;
  }


void TPoly :: GetPolyArea()
  {

   double AreaOneTri;
   double TotalArea = 0;
   double xNd1, yNd1, zNd1;
   double xNd2, yNd2, zNd2;
   double xNd3, yNd3, zNd3;
   double PolyArea;
   int StartTri, LastTri;

   cout << "Enter Begin Triangle #: " << endl;
   cin >> StartTri;
   cout << "Enter End Triangle #: " << endl;
   cin >> LastTri;

   Node.GetXYZCoordinates();

   for (int t = StartTri; t <= LastTri; t ++)
     {
      printf("\r");
```

```
      cout << "calculating area for TIN #: " << t;
      GetTINNodes(t, xNd1, yNd1, zNd1,
                     xNd2, yNd2, zNd2,
                     xNd3, yNd3, zNd3);
      AreaOneTri = GetTINArea(xNd1, yNd1, xNd2, yNd2, xNd3, yNd3);
      cout << endl;
      cout << "Area of Triangle: " << AreaOneTri << endl;
      PolyArea = PolyArea + AreaOneTri;
     }

   cout << endl;
   cout << "Total area of TINs: " << PolyArea << endl;
   //PressAnyKey();
  }

//*************************************************************//
//  Project: TIN-based Spatial Modelling using POET            //
//  File   : tsolid.cpp                                        //
//  Author : Alias Abdul-Rahman (c) Feb., 2000                 //
//                                                             //
//*************************************************************//

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <fstream.h>
#include <conio.h>

// POET includes
#include <poet.hxx>

// PTXX generated files
#include "tsolid.hxx"
#include "tnode.hxx"
#include "tpoly.hxx"

// Borland C++ includes
//#include "tsolid.h"
//#include "tpoly.h"
//#include "tnode.h"

#define maxpoint 10000
#define maxtriangle 10000


TNode Node;
TPoly Poly;

// the Implementation for the TSolid class
TSolid :: TSolid()       // constructor
  {

  }

TSolid :: ~TSolid()      // destructor
  {

  }
```

```
void TSolid :: PressAnyKey()
  {
   cout << endl;
   cout << endl;
   cout << "Press any key to continue" << endl;
   getch();
   cout << endl;
  }

void TSolid :: ReadTENs()
  {
   string charNode1, charNode2, charNode3, charNode4;

   // Reads the 3D TIN file
   ifstream TENfile = "c:\\data\\lakebo.ten";

   if (!TENfile)
     cout << "Unable to open file: TEN file" << endl;

   else
     {
      TENfile >> charNode1 >> charNode2 >> charNode3 >> charNode4;
      int nten = 0;
      do
        {
         nten ++;
         printf("\r");
         cout << "reading TENs ... " << nten;
         TENfile >> TEN[nten].Node1
                 >> TEN[nten].Node2
                 >> TEN[nten].Node3
                 >> TEN[nten].Node4;
        } while (!TENfile.eof());
     }

   cout << endl;
   cout << endl;
   TENfile.close();
  }


void TSolid :: Get3TINNodes(int t,
                            double& xNd1, double& yNd1, double& zNd1,
                            double& xNd2, double& yNd2, double& zNd2,
                            double& xNd3, double& yNd3, double& zNd3,
                            double& xNd4, double& yNd4, double& zNd4)
  {
   int Nd1, Nd2, Nd3, Nd4;

   Nd1 = TEN[t].Node1;
   Nd2 = TEN[t].Node2;
   Nd3 = TEN[t].Node3;
   Nd4 = TEN[t].Node4;

   xNd1 = Node.Point[Nd1].x;
   yNd1 = Node.Point[Nd1].y;
   zNd1 = Node.Point[Nd1].z;

   xNd2 = Node.Point[Nd2].x;
   yNd2 = Node.Point[Nd2].y;
   zNd2 = Node.Point[Nd2].z;
```

```
   xNd3 = Node.Point[Nd3].x;
   yNd3 = Node.Point[Nd3].y;
   zNd3 = Node.Point[Nd3].z;

   xNd4 = Node.Point[Nd4].x;
   yNd4 = Node.Point[Nd4].y;
   zNd4 = Node.Point[Nd4].z;
  }

double TSolid :: ComputeTENVolume(double x1, double y1, double z1,
                                  double x2, double y2, double z2,
                                  double x3, double y3, double z3,
                                  double x4, double y4, double z4)
  {
   double x41, y41, z41;
   double x31, y31, z31;
   double x21, y21, z21;
   double d32, d42, d43;

   x41 = x4 - x1;
   y41 = y4 - y1;
   z41 = z4 - z1;

   x31 = x3 - x1;
   y31 = y3 - y1;
   z31 = z3 - z1;

   x21 = x2 - x1;
   y21 = y2 - y1;
   z21 = z2 - z1;

   d32 = y31*z21 - z31*y21;
   d42 = y41*z21 - z41*y21;
   d43 = y41*z31 - z41*y31;

   return abs(0.166667 * (x41*d32 - x31*d42 + x21*d43));
  }

void TSolid :: GetVolume()
  {
   double VolOneTEN;
   double TotalVolume;
   int TEN1, TEN2;

   Node.GetBoreholeCoordinates();
   cout << "Enter range of TEN #: " << endl;
   cout << "         first TEN #: " << endl;
   cin >> TEN1;
   cout << "         last  TEN #: " << endl;
   cin >> TEN2;

   for (int t = TEN1; t <= TEN2; t ++)
     {
      Get3TINNodes(t, xNd1, yNd1, zNd1,
                      xNd2, yNd2, zNd2,
                      xNd3, yNd3, zNd3,
                      xNd4, yNd4, zNd4);

      VolOneTEN = ComputeTENVolume(xNd1, yNd1, zNd1,
                                   xNd2, yNd2, zNd2,
                                   xNd3, yNd3, zNd3,
```

```
                                    xNd4, yNd4, zNd4);
     TotalVolume = TotalVolume + VolOneTEN;
    }
 cout << endl;
 cout << "Total volume of the given TENs: " << TotalVolume << endl;
 //PressAnyKey();
}
```

User Interface (MDI) Code

```
//****************************************************************//
//  Project: User Interface (MDI)for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingapp.h
//
//
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Class definition for GraphingApp (TApplication).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//****************************************************************//

#if !defined(graphingapp_h)        // Sentry, use file only if it's not
#define graphingapp_h              //   already included.

#include <owl/controlb.h>
#include <owl/docking.h>
#include <owl/printer.h>
#include <owl/rcntfile.h>

#include "graphingmdiclient.h"

#include "graphingapp.rh"          // Definition of all resources.


//{{TApplication = GraphingApp}}
class GraphingApp : public TApplication, public TRecentFiles {
  private:
    void SetupSpeedBar(TDecoratedMDIFrame* frame);

  public:
    GraphingApp();
    virtual ~GraphingApp();

    void CreateGadgets(TDockableControlBar* cb, bool server = false);
    THarbor*        ApxHarbor;

    TGraphingMDIClient* MdiClient;

    // Public data members used by the print menu commands
    // and Paint routine in MDIChild.
    //
    TPrinter*       Printer;                // Printer support.
    int             Printing;               // Printing in progress.
```

```
//{{GraphingAppVIRTUAL_BEGIN}}
  public:
    virtual void InitMainWindow();
//{{GraphingAppVIRTUAL_END}}

//{{GraphingAppRSP_TBL_BEGIN}}
  protected:
    void EvNewView(TView& view);
    void EvCloseView(TView& view);
    void CmHelpAbout();
    void EvWinIniChange(char far* section);
    void EvOwlDocument(TDocument& doc);
    int32 CmFileSelected(uint wp, int32 lp);
//{{GraphingAppRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(GraphingApp);
};    //{{GraphingApp}}


#endif  // graphingapp_h sentry.
```

```
//************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TIN
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingapp.cpp
//
//
//  OVERVIEW
//
//  Source file for implementation of TGraphingApp.
//
//
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//************************************************************************//

#include <owl/pch.h>

#include <owl/buttonga.h>
#include <owl/statusba.h>
#include <owl/docmanag.h>
#include <owl/filedoc.h>

#include "graphingapp.h"
#include "graphingmdiclient.h"
#include "graphingmdichild.h"
#include "graphingwindowview.h"
#include "graphingaboutdlg.h"              // Definition of about dialog.
#include "graphdocument.h"                 // Definition of about dialog.

//{{GraphingApp Implementation}}



//{{DOC_VIEW}}
DEFINE_DOC_TEMPLATE_CLASS(GraphDocument, TGraphingWindowView, DocType1);

//{{DOC_VIEW_END}}

//{{DOC_MANAGER}}
DocType1 __dvt1("TIN Files", "*.xyz", 0, "*.xyz", dtAutoDelete | dtUpdateDir
                                                                        |
dtOverwritePrompt);

//{{DOC_MANAGER_END}}


//
//  Build a response table for all messages/commands handled by the
application.
//
DEFINE_RESPONSE_TABLE2(GraphingApp, TRecentFiles, TApplication)
//{{GraphingAppRSP_TBL_BEGIN}}
  EV_OWLVIEW(dnCreate, EvNewView),
  EV_OWLVIEW(dnClose,  EvCloseView),
  EV_COMMAND(CM_HELPABOUT, CmHelpAbout),
  EV_WM_WININICHANGE,
  EV_OWLDOCUMENT(dnCreate, EvOwlDocument),
```

```
  EV_OWLDOCUMENT(dnRename, EvOwlDocument),
  EV_REGISTERED(MruFileMessage, CmFileSelected),
//{{GraphingAppRSP_TBL_END}}
END_RESPONSE_TABLE;


//----------------------------------------------------------
// GraphingApp
// ~~~~~
//
GraphingApp::GraphingApp() : TApplication("TINSoft ver. 1.0"),
                                          TRecentFiles(".\\Graphing.ini", 4)
{
  Printer = 0;
  Printing = 0;

  SetDocManager(new TDocManager(dmMDI, this));

  // INSERT>> Your constructor code here.
}


GraphingApp::~GraphingApp()
{
  delete Printer;

  // INSERT>> Your destructor code here.
}


void GraphingApp::CreateGadgets(TDockableControlBar* cb, bool server)
{
  if (!server) {
    cb->Insert(*new TButtonGadget(CM_MDIFILENEW, CM_MDIFILENEW));
    cb->Insert(*new TButtonGadget(CM_MDIFILEOPEN, CM_MDIFILEOPEN));
    cb->Insert(*new TButtonGadget(CM_FILESAVE, CM_FILESAVE));
    cb->Insert(*new TSeparatorGadget(6));
  }

  //cb->Insert(*new TButtonGadget(CM_EDITCUT, CM_EDITCUT));
  //cb->Insert(*new TButtonGadget(CM_EDITCOPY, CM_EDITCOPY));
  //cb->Insert(*new TButtonGadget(CM_EDITPASTE, CM_EDITPASTE));
  //cb->Insert(*new TSeparatorGadget(6));
  //cb->Insert(*new TButtonGadget(CM_EDITUNDO, CM_EDITUNDO));
  //cb->Insert(*new TSeparatorGadget(6));
  //cb->Insert(*new TButtonGadget(CM_EDITFIND, CM_EDITFIND));
  //cb->Insert(*new TButtonGadget(CM_EDITFINDNEXT, CM_EDITFINDNEXT));

  if (!server) {
    cb->Insert(*new TSeparatorGadget(6));
    cb->Insert(*new TButtonGadget(CM_FILEPRINT, CM_FILEPRINT));
   cb->Insert(*new TButtonGadget(CM_FILEPRINTPREVIEW, CM_FILEPRINTPREVIEW));
  }

  // Add caption and fly-over help hints.
  //
  cb->SetCaption("Toolbar");
  cb->SetHintMode(TGadgetWindow::EnterHints);
}
```

```
void GraphingApp::SetupSpeedBar(TDecoratedMDIFrame* frame)
{
  ApxHarbor = new THarbor(*frame);

  // Create default toolbar New and associate toolbar buttons with commands.
  //
  TDockableControlBar* cb = new TDockableControlBar(frame);
  CreateGadgets(cb);

  // Setup the toolbar ID used by OLE 2 for toolbar negotiation.
  //
  cb->Attr.Id = IDW_TOOLBAR;

  ApxHarbor->Insert(*cb, alTop);
}


//----------------------------------------------------------
// GraphingApp
// ~~~~~
// Application main window construction & intialization.
//
void GraphingApp::InitMainWindow()
{
  if (nCmdShow != SW_HIDE)
    nCmdShow = (nCmdShow != SW_SHOWMINNOACTIVE) ? SW_SHOWNORMAL : nCmdShow;

  MdiClient = new TGraphingMDIClient(this);
  TDecoratedMDIFrame* frame = new TDecoratedMDIFrame(Name, IDM_MDI,
                                                     *MdiClient, true, this);

  nCmdShow = (nCmdShow != SW_SHOWMINNOACTIVE) ? SW_SHOWNORMAL : nCmdShow;

  // Assign icons for this application.
  //
  frame->SetIcon(this, IDI_MDIAPPLICATION);
  frame->SetIconSm(this, IDI_MDIAPPLICATION);

  // Associate with the accelerator table.
  //
  frame->Attr.AccelTable = IDM_MDI;

  TStatusBar* sb = new TStatusBar(frame, TGadget::Recessed,
                                  TStatusBar::CapsLock        |
                                  TStatusBar::NumLock         |
                                  TStatusBar::ScrollLock);
  frame->Insert(*sb, TDecoratedFrame::Bottom);

  SetupSpeedBar(frame);

  SetMainWindow(frame);

  frame->SetMenuDescr(TMenuDescr(IDM_MDI));

}


//----------------------------------------------------------
// GraphingApp
// ~~~~~
```

```
// Response Table handlers:
//
void GraphingApp::EvNewView(TView& view)
{
          T M D I C l i e n t *      m d i C l i e n t      =
TYPESAFE_DOWNCAST(GetMainWindow()->GetClientWindow(),
                                                          TMDIClient);
  if (mdiClient) {
    TGraphingMDIChild* child = new TGraphingMDIChild(*mdiClient, 0,
                                                     view.GetWindow());

    // Set the menu descriptor for this view.
    //
    if (view.GetViewMenu())
      child->SetMenuDescr(*view.GetViewMenu());

    // Assign icons with this child window.
    //
    child->SetIcon(this, IDI_DOC);
    child->SetIconSm(this, IDI_DOC);

    child->Create();
  }
}


void GraphingApp::EvCloseView(TView&)
{
}


//----------------------------------------------------------
// GraphingApp
// ~~~~~~~~~~~
// Menu Help About Graphing command
void GraphingApp::CmHelpAbout()
{
  // Show the modal dialog.
  //
  GraphingAboutDlg(GetMainWindow()).Execute();
}


void GraphingApp::EvOwlDocument(TDocument& doc)
{
  if (doc.GetDocPath())
    SaveMenuChoice(doc.GetDocPath());
}


int32 GraphingApp::CmFileSelected(uint wp, int32)
{
  TAPointer<char> text = new char[_MAX_PATH];

  GetMenuText(wp, text, _MAX_PATH);
  TDocTemplate* tpl = GetDocManager()->MatchTemplate(text);
  if (tpl)
    GetDocManager()->CreateDoc(tpl, text);
  return 0;
}
```

```
void GraphingApp::EvWinIniChange(char far* section)
{
  if (strcmp(section, "windows") == 0) {
    // If the device changed in the WIN.INI file then the printer
    // might have changed.  If we have a TPrinter(Printer) then
    // check and make sure it's identical to the current device
    // entry in WIN.INI.
    //
    if (Printer) {
      const int bufferSize = 255;
      char printDBuffer[bufferSize];
      LPSTR printDevice = printDBuffer;
      LPSTR devName;
      LPSTR driverName = 0;
      LPSTR outputName = 0;
           if  (::GetProfileString("windows",  "device",  "",  printDevice,
bufferSize)) {
        // The string which should come back is something like:
        //
        //      HP LaserJet III,hppcl5a,LPT1:
        //
        // Where the format is:
        //
        //      devName,driverName,outputName
        //
        devName = printDevice;
        while (*printDevice) {
          if (*printDevice == ',') {
            *printDevice++ = 0;
            if (!driverName)
              driverName = printDevice;
            else
              outputName = printDevice;
          }
          else
            printDevice = ::AnsiNext(printDevice);
        }

        if (Printer->GetSetup().Error != 0                    ||
            strcmp(devName, Printer->GetSetup().GetDeviceName()) != 0  ||
            strcmp(driverName, Printer->GetSetup().GetDriverName()) != 0 ||
            strcmp(outputName, Printer->GetSetup().GetOutputName()) != 0 ) {
          // New printer installed so get the new printer device now.
          //
          delete Printer;
          Printer = new TPrinter(this);
        }
      }
      else {
        // No printer installed(GetProfileString failed).
        //
        delete Printer;
        Printer = new TPrinter(this);
      }
    }
  }
}


int OwlMain(int , char* [])
{
```

```
  GraphingApp   app;
  return app.Run();
}
```

```
//************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    graphing.apx Application
//
//  FILE:         graphdocument.h
//
//
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Class definition for GraphDocument (TFileDocument).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//************************************************************************//

#if !defined(graphdocument_h)  // Sentry, use file only if it's not already
#define graphdocument_h        //    included.

#include <owl/filedoc.h>

#include "graphingapp.rh"            // Definition of all resources.
#include <list>
#include <vector>


      struct Point
        {
         float x;
         float y;
         float z;
         int operator ==(const Point& p)
             const
                {
                 return x == p.x &&
                        y == p.y &&
                        z == p.z;
                }

         int operator <(const Point& p)
             const
               {
                return x < p.x &&
                       y < p.y &&
                       z < p.z;
               }
        };

      struct Triangle
        {
         int node_1;
         int node_2;
         int node_3;
         int operator ==(const Triangle& t)
             const
               {
                return node_1 == t.node_1 &&
                       node_2 == t.node_2 &&
```

```
                    node_3 == t.node_3;
                }

        int operator <(const Triangle& t)
            const
              {
               return node_1 < t.node_1 &&
                      node_2 < t.node_2 &&
                      node_3 < t.node_3;
                }
        };


    struct Segment
       {
        float x;
        float y;
        int Hreq;
        int SegNr;
        int operator ==(const Segment& seg)
            const
              {
               return x     == seg.x    &&
                      y     == seg.y    &&
                      Hreq  == seg.Hreq &&
                      SegNr == seg.SegNr;
                }

        int operator <(const Segment& seg)
            const
              {
               return x     < seg.x    &&
                      y     < seg.y    &&
                      Hreq  < seg.Hreq &&
                      SegNr < seg.SegNr;
                }
        };


//{{TFileDocument = GraphDocument}}
class GraphDocument : public TFileDocument {
  public:
    GraphDocument(TDocument* parent = 0);
    virtual ~GraphDocument();

//{{GraphDocumentVIRTUAL_BEGIN}}
  public:
    void GetXYZ_TIN()
      {
       if (!IsOpen())
         Open(ofRead) ;
      }

    virtual bool Open(int mode, const char far* path=0);
//{{GraphDocumentVIRTUAL_END}}

// Use STL containers to make life easier for
//      points, triangles, contour segments, etc.
  std::vector<Point> points;
  std::list<Triangle> triangles;
```

```
  std::list<Segment> contours;

  float xmin, ymin, xmax, ymax, zmin, zmax;
  };     //{{GraphDocument}}


#endif  // graphdocument_h sentry.
```

```
//****************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    graphing.apx Application
//
//  FILE:         graphdocument.cpp
//
//
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Source file for implementation of GraphDocument (TFileDocument).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#include <owl/pch.h>
#include <string.h>

#include "graphdocument.h"


//{{GraphDocument Implementation}}


GraphDocument::GraphDocument(TDocument* parent)
:
    TFileDocument(parent)
{
  // INSERT>> Your constructor code here.
}


GraphDocument::~GraphDocument()
{
  // INSERT>> Your destructor code here.
  Close();
}


bool GraphDocument::Open(int mode, const char far* path)
{
  bool result;
  // Read in all the data once
   if (path)
    SetDocPath(path);
  if (GetDocPath()) {
    TInStream* ist = InStream(ofRead);
    if (!ist)
      return false;

    bool ftt = true;
    Point p;

    string charX, charY, charZ;
    // read the file header
    *ist >> charX >> charY >> charZ;
```

```
while (*ist) {
  *ist >> p.x >> p.y >> p.z;
  if (*ist) {
    points.push_back(p);
    // While we are here - get the min & max
    if (ftt)  {
      xmin = xmax = p.x;
      ymin = ymax = p.y;
      zmin = zmax = p.z;
      ftt = false;
    } else {
      xmin = min(xmin,p.x);
      ymin = min(ymin,p.y);
      zmin = min(zmin,p.z);
      xmax = max(xmax,p.x);
      ymax = max(ymax,p.y);
      zmax = max(zmax,p.z);
    }
  }
}

char t_path[MAX_PATH];
strcpy(t_path, GetDocPath());
// Nasty fudge to pick up matching TIN file
for (int i = strlen(t_path) - 1; i > 0; i--) {
  if (t_path[i] == '.') {
   t_path[i + 1] = 'T';
   t_path[i + 2] = 'I';
   t_path[i + 3] = 'N';
   break;
  }
}

// read the file header of the TIN file
string charNode1, charNode2 , charNode3;
ifstream in_t(t_path);

// Read the TINS
Triangle t;

in_t >> charNode1 >> charNode2 >> charNode3;

while (in_t) {
  in_t >> t.node_1 >> t.node_2 >> t.node_3;
  if (in_t)
    triangles.push_back(t);

}


// to read the contour segment file
char c_path[MAX_PATH];
 strcpy(c_path, GetDocPath());
 // Nasty fudge to pick up matching CON file
 for (int i = strlen(c_path) - 1; i > 0; i--) {
   if (c_path[i] == '.') {
    c_path[i + 1] = 'C';
    c_path[i + 2] = 'O';
    c_path[i + 3] = 'N';
    break;
   }
```

```
    }

    // read the header of the CON file
    string charXcon, charYcon, charHreq, charSegNr;
    ifstream in_c(c_path);

    // Read the Segments
    Segment seg;
    in_c >> charXcon >> charYcon >> charHreq >> charSegNr;

    while (in_c) {
      in_c >> seg.x >> seg.y >> seg.Hreq >> seg.SegNr;
      if (in_c)
        contours.push_back(seg);

    }

  delete ist;

  }
  return result;
}
```

```
//****************************************************************************//
//  Project: MDI for 2D/3D TINs
//
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         apxprint.h
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Class definition for TApxPrintout (TPrintout).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#if !defined(apxprint_h)    // Sentry use file only if it's not already
included.
#define apxprint_h

#include <owl/printer.h>


class TApxPrintout : public TPrintout {
  public:
    TApxPrintout(TPrinter* printer, const char far* title, TWindow* window,
                                                          bool scale = true)
      : TPrintout(title)
      {
        Printer = printer;
        Window = window;
        Scale = scale;
        MapMode = MM_ANISOTROPIC;
      }

    void GetDialogInfo(int& minPage, int& maxPage, int& selFromPage,
                       int& selToPage);
    void BeginPrinting();
    void BeginPage(TRect& clientR);
    void PrintPage(int page, TRect& rect, unsigned flags);
    void EndPage();
    void SetBanding(bool b)        { Banding = b; }
    bool HasPage(int pageNumber);

  protected:
    TWindow*  Window;
    bool      Scale;
    TPrinter* Printer;
    int       MapMode;

    int       PrevMode;
    TSize     OldVExt, OldWExt;
    TRect     OrgR;
};

#endif  // apxprint_h
```

```
//****************************************************************************//
//  Project: MDI for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         apxprint.cpp
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Source file for implementation of Printing.
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#include <owl/pch.h>
#include "apxprint.h"


// Do not enable page range in the print dialog since only one page is
// available to be printed
//
void TApxPrintout::GetDialogInfo(int& minPage, int&, int& selFromPage, int&)
{
  minPage = 1;
  selFromPage = 0;
}


void TApxPrintout::BeginPrinting()
{
  TRect clientR;

  BeginPage(clientR);

  HFONT   hFont = (HFONT)Window->GetWindowFont();
  TFont   font("Arial", -12);
  if (!hFont)
    DC->SelectObject(font);
  else
    DC->SelectObject(TFont(hFont));

  TEXTMETRIC  tm;
  int fHeight = DC->GetTextMetrics(tm) ? tm.tmHeight + tm.tmExternalLeading
: 10;

  DC->RestoreFont();

  // How many lines of this font can we fit on a page.
  //
  int linesPerPage = MulDiv(clientR.Height(), 1, fHeight);

  TPrintDialog::TData& printerData = Printer->GetSetup();

   // GetMinMaxInfo event is overrided to return the number of lines when
printing.
  //
  MINMAXINFO minmaxinfo;
  Window->SendMessage(WM_GETMINMAXINFO, 0, (long)&minmaxinfo);
```

```
  int maxPg = (minmaxinfo.ptMaxSize.y / linesPerPage) + 1.0;

  // Compute the number of pages to print.
  //
  printerData.MinPage = 1;
  printerData.MaxPage = maxPg;

  EndPage();

  TPrintout::BeginPrinting();
}


void TApxPrintout::BeginPage(TRect& clientR)
{
  TScreenDC screenDC;
  TSize screenRes(screenDC.GetDeviceCaps(LOGPIXELSX),
          screenDC.GetDeviceCaps(LOGPIXELSY));
  TSize printRes(DC->GetDeviceCaps(LOGPIXELSX),
           DC->GetDeviceCaps(LOGPIXELSY));

  // Temporarily change the window size (so any WM_PAINT queries on the total
  // window size (GetClientRect) is
  // the window size for the WM_PAINT of the window and the printer page size
  // when Paint is called from
  // PrintPage. Notice, we don't use AdjustWindowRect because its harder and
  // not accurate.  Instead we
  // compute the difference (in pixels) between the client window and the
  // frame window.  This difference
  // is then added to the clientRect to compute the new frame window size
  // for SetWindowPos.
  //
  clientR = Window->GetClientRect();
  Window->MapWindowPoints(HWND_DESKTOP, (TPoint*)&clientR, 2);

  // Compute extra X and Y pixels to bring a client window dimensions to
  // equal the frame window.
  //
  OrgR = Window->GetWindowRect();
  int adjX = OrgR.Width() - clientR.Width();
  int adjY = OrgR.Height() - clientR.Height();

  // Conditionally scale the DC to the window so the printout will
  // resemble the window.
  //
  if (Scale) {
    clientR = Window->GetClientRect();
    PrevMode = DC->SetMapMode(MapMode);
    DC->SetViewportExt(PageSize, &OldVExt);

    // Scale window to logical page size (assumes left & top are 0)
    //
    clientR.right = MulDiv(PageSize.cx, screenRes.cx, printRes.cx);
    clientR.bottom = MulDiv(PageSize.cy, screenRes.cy, printRes.cy);

    DC->SetWindowExt(clientR.Size(), &OldWExt);
  }

  // Compute the new size of the window based on the printer DC dimensions.
  // Resize the window, notice position, order, and redraw are not done
  // the window size changes but the user
```

```
  // doesn't see any visible change to the window.
  //
  Window->SetRedraw(false);
   Window->SetWindowPos(0, 0, 0, clientR.Width() + adjX, clientR.Height() +
adjY,
                                SWP_NOMOVE | SWP_NOREDRAW | SWP_NOZORDER|
SWP_NOACTIVATE);
}


void TApxPrintout::PrintPage(int page, TRect& bandRect, unsigned)
{
  TRect clientR;

  BeginPage(clientR);

  if (Scale)
    DC->DPtoLP(bandRect, 2);

  // Change the printer range to this current page.
  //
  TPrintDialog::TData& printerData = Printer->GetSetup();
  int fromPg = printerData.FromPage;
  int toPg = printerData.ToPage;

  printerData.FromPage = page;
  printerData.ToPage = page;

  // Call the window to paint itself to the printer DC.
  //
  Window->Paint(*DC, false, bandRect);

  printerData.FromPage = fromPg;
  printerData.ToPage = toPg;

  if (Scale)
    DC->LPtoDP(bandRect, 2);

  EndPage();
}


void TApxPrintout::EndPage()
{
  // Resize to original window size, no one's the wiser.
  //
  Window->SetWindowPos(0, 0, 0, OrgR.Width(), OrgR.Height(),
               SWP_NOMOVE | SWP_NOREDRAW | SWP_NOZORDER| SWP_NOACTIVATE);
  Window->SetRedraw(true);

  // Restore changes made to the DC
  //
  if (Scale) {
    DC->SetWindowExt(OldWExt);
    DC->SetViewportExt(OldVExt);
    DC->SetMapMode(PrevMode);
   }
}


bool TApxPrintout::HasPage(int pageNumber)
```

```
{
  TPrintDialog::TData& printerData = Printer->GetSetup();

      return pageNumber >= printerData.MinPage && pageNumber <=
printerData.MaxPage;
}
```

```
//****************************************************************************//
//  Project: MDI for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingwindowview.h
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Class definition for TGraphingWindowView (TWindowView).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#if !defined(graphingwindowview_h)    // Sentry, use file only if it's not
#define graphingwindowview_h          // ... already included.

#include <owl/docview.h>

#include "graphingapp.rh"             // Definition of all resources.


//{{TWindowView = TGraphingWindowView}}
class TGraphingWindowView : public TWindowView {
  public:
    TGraphingWindowView(TDocument& doc, TWindow* parent = 0);
    virtual ~TGraphingWindowView();

//{{TGraphingWindowViewVIRTUAL_BEGIN}}
  public:
    virtual void Paint(TDC& dc, bool erase, TRect& rect);
//{{TGraphingWindowViewVIRTUAL_END}}
//{{TGraphingWindowViewRSP_TBL_BEGIN}}
  protected:
    void EvGetMinMaxInfo(MINMAXINFO far& minmaxinfo);
    void EvSize(uint sizeType, TSize& size);
//{{TGraphingWindowViewRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TGraphingWindowView);
};    //{{TGraphingWindowView}}


#endif  // graphingwindowview_h sentry.
```

```
//****************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TIN
//
//  SUBSYSTEM:    Graphing Application                                         //
//  FILE:         graphingwindowview.cpp                                      //
//  OVERVIEW                                                                  //
//  ~~~~~~~~                                                                  //
//  Source file for implementation of TGraphingWindowView (TWindowView).      //
//  Author: Alias Abdul-Rahman (c) 1999                                       //
//****************************************************************************//

#include <owl/pch.h>

#include "graphingapp.h"
#include "graphingwindowview.h"
#include "graphdocument.h"

#include <stdio.h>


//{{TGraphingWindowView Implementation}}


//
// Build a response table for all messages/commands handled
// by TGraphingWindowView derived from TWindowView.
//
DEFINE_RESPONSE_TABLE1(TGraphingWindowView, TWindowView)
//{{TGraphingWindowViewRSP_TBL_BEGIN}}
  EV_WM_GETMINMAXINFO,
  EV_WM_SIZE,
//{{TGraphingWindowViewRSP_TBL_END}}
END_RESPONSE_TABLE;


//----------------------------------------------------------
// TGraphingWindowView
// ~~~~~~~~~~
// Construction/Destruction handling.
//
TGraphingWindowView::TGraphingWindowView(TDocument& doc, TWindow* parent)
  :
  TWindowView(doc, parent)
 {
  // INSERT>> Your constructor code here.
 }


TGraphingWindowView::~TGraphingWindowView()
 {
  // INSERT>> Your destructor code here.
 }


//
// Paint routine for Window, Printer, and PrintPreview for a TWindowView
client.
//
void TGraphingWindowView::Paint(TDC& dc, bool, TRect& rect)
 {
  HDC hdc;
```

```
  // to set the back ground color,
  // comment out the next line for default color
  //static const TColor color (RGB(192, 192, 192)); // RGB - light gray
  //TWindow::SetBkgndColor(color);
  GraphingApp* theApp = TYPESAFE_DOWNCAST(GetApplication(), GraphingApp);
  if (theApp)
    {
    // Only paint if we're printing and we have something to paint,
    // otherwise do nothing.
    //
    if (theApp->Printing && theApp->Printer && !rect.IsEmpty()) {
      // Use pageSize to get the size of the window to render into.
      // For a Window it's the client area,
      // for a printer it's the printer DC dimensions and for
      // print preview it's the layout window.
      //
      TSize   pageSize(rect.right - rect.left, rect.bottom - rect.top);

      TPrintDialog::TData& printerData = theApp->Printer->GetSetup();

      // Compute the number of pages to print.
      //
      printerData.MinPage = 1;
      printerData.MaxPage = 1;

      // INSERT>> Special printing code goes here.

    }
    else
     {
      // INSERT>> Normal painting code goes here.
     }
      GraphDocument& g_doc = dynamic_cast<GraphDocument&>(GetDocument());
      // Make sure the doc has read the points
      g_doc.GetXYZ_TIN();

      /*
      // Set the mapping mode so that all points fit on the scale
      dc.SetMapMode(MM_ANISOTROPIC);
            dc.SetWindowExt(TSize((g_doc.xmax  -  g_doc.xmin),(g_doc.ymax  -
g_doc.ymin)));
      dc.SetWindowOrg(TPoint(g_doc.xmin,g_doc.ymin));
      dc.SetViewportExt(GetClientRect().Size());
      dc.SetViewportOrg(TPoint(0, 0));
      */

      // set the device origin
      SetViewportOrgEx(dc, rect.left, rect.top, &POINT());

      float xlength = g_doc.xmax - g_doc.xmin;
      float ylength = g_doc.ymax - g_doc.ymin;

      float scaleX = (rect.right - rect.left) / xlength;
      float scaleY = (rect.bottom - rect.top) / ylength;


    // to handle points only
    TPen penpoints = TPen(RGB(0, 0, 255), 2, PS_SOLID);
    dc.SelectObject(penpoints);

    char s[16];
```

```
    // Use STL container to go through container in read order
    for (std::vector<Point>::iterator i = g_doc.points.begin();
                                        i != g_doc.points.end() ; i++)
     {
      int x = ((*i).x - g_doc.xmin) * scaleX;
      int y = (rect.bottom - rect.top) - ((*i).y - g_doc.ymin) * scaleY;
      int z = (*i).z;

      dc.MoveTo(x, y);
      dc.LineTo(x, y);
     }

// to handle triangles
#define TRIANGLES
// Comment out the above line to see the contours

#if defined TRIANGLES
     TPen pentri = TPen(RGB(0, 0, 255), 1, PS_SOLID);
     dc.SelectObject(pentri);

     for (std::list<Triangle>::iterator i = g_doc.triangles.begin();
                                          i != g_doc.triangles.end(); i++)
      {
       int xNd1 = (g_doc.points[(*i).node_1-1].x - g_doc.xmin) * scaleX;
       int yNd1 = (rect.bottom - rect.top) - (g_doc.points[(*i).node_1-1].y
                                           - g_doc.ymin) * scaleY;
       int xNd2 = (g_doc.points[(*i).node_2-1].x - g_doc.xmin) * scaleX;
       int yNd2 = (rect.bottom - rect.top) - (g_doc.points[(*i).node_2-1].y
                                           - g_doc.ymin) * scaleY;
       int xNd3 = (g_doc.points[(*i).node_3-1].x - g_doc.xmin) * scaleX;
       int yNd3 = (rect.bottom - rect.top) - (g_doc.points[(*i).node_3-1].y
                                           - g_doc.ymin) * scaleY;

       dc.MoveTo(xNd1, yNd1);
       dc.LineTo(xNd2, yNd2);
       dc.LineTo(xNd3, yNd3);
       dc.LineTo(xNd1, yNd1);
      }

#endif


// to handle Contour segments
#define CONTOURS
// Comment out the above line to see the contours

#if defined CONTOURS
     TPen pencont = TPen(RGB(255, 0, 0), 1, PS_SOLID);
     dc.SelectObject(pencont);

     int x = ((*g_doc.contours.begin()).x - g_doc.xmin) * scaleX;
     int y = (rect.bottom - rect.top) - ((*g_doc.contours.begin()).y
                                          - g_doc.ymin) * scaleY;
     int Hreq = (*g_doc.contours.begin()).Hreq;
     int SegNr = (*g_doc.contours.begin()).SegNr;
     int prev_Hreq = Hreq;
     int prev_SegNr = SegNr;
     int startx = x;
     int starty = y;
     dc.MoveTo(x, y);
```

```
    for (std::list<Segment>::iterator i = g_doc.contours.begin();
                                        i != g_doc.contours.end(); i++)
     {
      int nextx = ((*i).x - g_doc.xmin) * scaleX;
      int nexty = (rect.bottom - rect.top) - ((*i).y - g_doc.ymin) * scaleY;

      int nextHreq = (*i).Hreq;
      int nextSegNr = (*i).SegNr;

      if (SegNr != prev_SegNr)
        {
         dc.MoveTo(x, y);
         startx = x;
         starty = y;
        }
      else
        {
         if ((x == startx) && (y == starty))
            {
             if (SegNr == nextSegNr)
                dc.MoveTo(x, y);
             else
                dc.LineTo(x, y);
            }
         else
         dc.LineTo(x, y);
        }

      prev_SegNr = SegNr;
      SegNr = nextSegNr;

      x = nextx;
      y = nexty;
      Hreq = nextHreq;
     }
    dc.MoveTo(x, y);

#endif

    dc.RestorePen();
   }
 }


void TGraphingWindowView::EvGetMinMaxInfo(MINMAXINFO far& minmaxinfo)
 {
  GraphingApp* theApp = TYPESAFE_DOWNCAST(GetApplication(), GraphingApp);
  if (theApp)
    {
     if (theApp->Printing)
       {
        minmaxinfo.ptMaxSize = TPoint(32000, 32000);
        minmaxinfo.ptMaxTrackSize = TPoint(32000, 32000);
        return;
       }
    }
  TWindowView::EvGetMinMaxInfo(minmaxinfo);
 }

void TGraphingWindowView::EvSize(uint sizeType, TSize& size)
 {
```

```
    TWindowView::EvSize(sizeType, size);

    // INSERT>> Your code here.
    Invalidate();
}
```

```
//////////////////////////////////////////////////////////////////////////
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingmdiclient.h
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Class definition for TGraphingMDIClient (TMDIClient).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//////////////////////////////////////////////////////////////////////////

#if !defined(graphingmdiclient_h)     // Sentry, use file only if it's not
#define graphingmdiclient_h           // already included

#include "graphingapp.rh"             // Definition of all resources.


//{{TMDIClient = TGraphingMDIClient}}
class TGraphingMDIClient : public TMDIClient {
  public:
    int      ChildCount;                  // Number of child window created.

    TGraphingMDIClient(TModule* module = 0);
    virtual ~TGraphingMDIClient();

    void OpenFile(const char* fileName = 0);

//{{TGraphingMDIClientVIRTUAL_BEGIN}}
  protected:
    virtual void SetupWindow();
//{{TGraphingMDIClientVIRTUAL_END}}

//{{TGraphingMDIClientRSP_TBL_BEGIN}}
  protected:
    void CmFilePrint();
    void CmFilePrintSetup();
    void CmFilePrintPreview();
    void CmPrintEnable(TCommandEnabler& tce);
//{{TGraphingMDIClientRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TGraphingMDIClient);
};    //{{TGraphingMDIClient}}


#endif  // graphingmdiclient_h sentry.
```

```
//****************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingmdiclient.cpp
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Source file for implementation of TGraphingMDIClient (TMDIClient).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#include <owl/pch.h>

#include <owl/docmanag.h>
#include <owl/listbox.h>
#include <stdio.h>

#include "graphingapp.h"
#include "graphingmdichild.h"
#include "graphingmdiclient.h"
#include "apxprint.h"
#include "apxprev.h"


//{{TGraphingMDIClient Implementation}}


//
// Build a response table for all messages/commands handled
// by TGraphingMDIClient derived from TMDIClient.
//
DEFINE_RESPONSE_TABLE1(TGraphingMDIClient, TMDIClient)
//{{TGraphingMDIClientRSP_TBL_BEGIN}}
  EV_COMMAND(CM_FILEPRINT, CmFilePrint),
  EV_COMMAND(CM_FILEPRINTERSETUP, CmFilePrintSetup),
  EV_COMMAND(CM_FILEPRINTPREVIEW, CmFilePrintPreview),
  EV_COMMAND_ENABLE(CM_FILEPRINT, CmPrintEnable),
  EV_COMMAND_ENABLE(CM_FILEPRINTERSETUP, CmPrintEnable),
  EV_COMMAND_ENABLE(CM_FILEPRINTPREVIEW, CmPrintEnable),
//{{TGraphingMDIClientRSP_TBL_END}}
END_RESPONSE_TABLE;


//----------------------------------------------------------
// TGraphingMDIClient
// ~~~~~~~~~~~~
// Construction/Destruction handling.
//
TGraphingMDIClient::TGraphingMDIClient(TModule* module)
:
  TMDIClient(module)
{
  ChildCount = 0;

  // INSERT>> Your constructor code here.
```

```
}


TGraphingMDIClient::~TGraphingMDIClient()
{
  Destroy();

  // INSERT>> Your destructor code here.

}


//----------------------------------------------------------
// TGraphingMDIClient
// ~~~~~~~~~~~
// MDIClient site initialization.
//
void TGraphingMDIClient::SetupWindow()
{
  // Default SetUpWindow processing.
  //
  TMDIClient::SetupWindow();

}


//----------------------------------------------------------
// TGraphingMDIClient
// ~~~~~~~~~~
// Menu File Print command
//
void TGraphingMDIClient::CmFilePrint()
{
  // Create Printer object if not already created.
  //
  GraphingApp* theApp = TYPESAFE_DOWNCAST(GetApplication(), GraphingApp);
  if (theApp) {
    if (!theApp->Printer)
      theApp->Printer = new TPrinter(theApp);

    TAPointer<char> docName = new char[_MAX_PATH];

    TDocument* currentDoc = theApp->GetDocManager()->GetCurrentDoc();
    if (currentDoc->GetTitle())
      strcpy(docName, currentDoc->GetTitle());
    else
      strcpy(docName, "Document");

    // Create Printout window and set characteristics.
    //
    TApxPrintout printout(theApp->Printer, docName,
                          GetActiveMDIChild()->GetClientWindow(), true);

    theApp->Printing++;

    // Bring up the Print dialog and print the document.
    //
    theApp->Printer->Print(GetWindowPtr(GetActiveWindow()), printout, true);

    theApp->Printing--;
  }
```

```
}


//----------------------------------------------------------
// TGraphingMDIClient
// ~~~~~~~~~~
// Menu File Print Setup command
//
void TGraphingMDIClient::CmFilePrintSetup()
{
  GraphingApp* theApp = TYPESAFE_DOWNCAST(GetApplication(), GraphingApp);
  if (theApp) {
    if (!theApp->Printer)
      theApp->Printer = new TPrinter(theApp);

    // Bring up the Print Setup dialog.
    //
    theApp->Printer->Setup(this);
  }
}


//----------------------------------------------------------
// TGraphingMDIClient
// ~~~~~~~~~~
// Menu File Print Preview command
//
void TGraphingMDIClient::CmFilePrintPreview()
{
  GraphingApp* theApp = TYPESAFE_DOWNCAST(GetApplication(), GraphingApp);
  if (theApp) {
    if (!theApp->Printer)
      theApp->Printer = new TPrinter(GetApplication());

    theApp->Printing++;

    TApxPreviewWin* prevW = new TApxPreviewWin(Parent, theApp->Printer,
                            GetActiveMDIChild()->GetClientWindow(),
                            "Print Preview", new TLayoutWindow(0));
    prevW->Create();

    // Here we resize the preview window to take the size of the MainWindow,
    // then hide the MainWindow.
    //
    TFrameWindow * mainWindow =  GetApplication()->GetMainWindow();
    TRect r = mainWindow->GetWindowRect();
    prevW->MoveWindow(r);
    prevW->ShowWindow(SW_SHOWNORMAL);
    mainWindow->ShowWindow(SW_HIDE);

    GetApplication()->BeginModal(GetApplication()->GetMainWindow());

    // After the user closes the preview Window, we take its size and use it
    // to size the MainWindow, then show the MainWindow again.
    //
    r = prevW->GetWindowRect();
    mainWindow->MoveWindow(r);
    mainWindow->ShowWindow(SW_SHOWNORMAL);

    // We must destroy the preview window explicitly.  Otherwise, the window
will
```

```
    // not be destroyed until it's parent the MainWindow is destroyed.
    //
    prevW->Destroy();
    delete prevW;

    theApp->Printing--;
  }
}


//----------------------------------------------------------
// TGraphingMDIClient
// ~~~~~~~~~~
// Menu enabler used by Print, Print Setup and Print Preview.
//
void TGraphingMDIClient::CmPrintEnable(TCommandEnabler& tce)
{
  if (GetActiveMDIChild()) {
    GraphingApp* theApp = TYPESAFE_DOWNCAST(GetApplication(), GraphingApp);
    if (theApp) {
      // If we have a Printer already created just test if all is okay.
      // Otherwise create a Printer object and make sure the printer really
      // exists and then delete the Printer object.
      //
      if (!theApp->Printer) {
        theApp->Printer = new TPrinter(theApp);
        tce.Enable(!theApp->Printer->GetSetup().Error);
      }
      else
        tce.Enable(!theApp->Printer->GetSetup().Error);
    }
  }
  else
    tce.Enable(false);
}
```

```
//****************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingmdichild.h
//
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Class definition for TGraphingMDIChild (TMDIChild).
//
//  Author : Alias Abdul-Rahman  (c) 1999
//
//****************************************************************************//

#if !defined(graphingmdichild_h)      // Sentry, use file only if it's not
#define graphingmdichild_h            // not alraedy included

#include "graphingapp.rh"             // Definition of all resources.


//{{TMDIChild = TGraphingMDIChild}}
class TGraphingMDIChild : public TMDIChild {
  public:
    TGraphingMDIChild(TMDIClient& parent, const char far* title,
                      TWindow* clientWnd, bool shrinkToClient = false,
                      TModule* module = 0);
    virtual ~TGraphingMDIChild();
};    //{{TGraphingMDIChild}}


#endif  // graphingmdichild_h sentry.
```

```
//****************************************************************************//
//  Project: MDI for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingmdichild.cpp
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Source file for implementation of TGraphingMDIChild (TMDIChild).
//
//  Author : Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#include <owl/pch.h>

#include "graphingapp.h"
#include "graphingmdichild.h"


//{{TGraphingMDIChild Implementation}}


//----------------------------------------------------------
// TGraphingMDIChild
// ~~~~~~~~~~
// Construction/Destruction handling.
//
TGraphingMDIChild::TGraphingMDIChild(TMDIClient& parent,
                                     const char far* title, TWindow*
clientWnd,
                                     bool shrinkToClient, TModule* module)
:
  TMDIChild(parent, title, clientWnd, shrinkToClient, module)
{
  // INSERT>> Your constructor code here.

}


TGraphingMDIChild::~TGraphingMDIChild()
{
  Destroy();

  // INSERT>> Your destructor code here.

}
```

```
//************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application                                     //
//  FILE:         apxprev.h
//
//  OVERVIEW                                                              //
//  ~~~~~~~~                                                              //
//  Class definition for TApxPreviewWin (Print Preview).                  //
//  Author: Alias Abdul-Rahman (c) 1999                                   //
//************************************************************************//

#if !defined(apxprev_h)     // Sentry, use file only if it's not already
included.
#define apxprev_h

#include <owl/controlb.h>
#include <owl/preview.h>

#include "apxprint.h"
#include "graphingapp.rh"


//{{TDecoratedFrame = TApxPreviewWin}}
class TApxPreviewWin : public TDecoratedFrame {
  public:
       TApxPreviewWin(TWindow* parentWindow, TPrinter* printer, TWindow*
currWindow,
                      const char far* title, TLayoutWindow* client);
   ~TApxPreviewWin();

    int             PageNumber, FromPage, ToPage;

    TWindow*        CurrWindow;
    TControlBar*    PreviewSpeedBar;
    TPreviewPage*   Page1;
    TPreviewPage*   Page2;
    TPrinter*       Printer;

    TPrintDC*       PrnDC;
    TSize*          PrintExtent;
    TApxPrintout*   Printout;

  private:
    TLayoutWindow*  Client;

    void SpeedBarState();
    void PPR_PreviousEnable(TCommandEnabler& tce);
    void PPR_NextEnable(TCommandEnabler& tce);
    void PPR_Previous();
    void PPR_Next();
    void PPR_OneUp();
    void PPR_TwoUpEnable(TCommandEnabler& tce);
    void PPR_TwoUp();
    void PPR_Done();
    void CmPrintEnable(TCommandEnabler& tce);
    void CmPrint();

//{{TApxPreviewWinVIRTUAL_BEGIN}}
  protected:
    virtual void SetupWindow();
```

```
//{{TApxPreviewWinVIRTUAL_END}}

//{{TApxPreviewWinRSP_TBL_BEGIN}}
  protected:
    void EvClose();
//{{TApxPreviewWinRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TApxPreviewWin);
};    //{{TApxPreviewWin}}

#endif  // apxprev_h
```

```
//****************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         apxprev.cpp
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Source file for implementation of Print Preview.
//
//  Author : Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#include <owl/pch.h>

#include <owl/buttonga.h>
#include <owl/textgadg.h>
#include <stdio.h>

#include "apxprev.h"
#include "graphingapp.rh"


//{{TApxPreviewWin Implementation}}

DEFINE_RESPONSE_TABLE1(TApxPreviewWin, TDecoratedFrame)
  EV_COMMAND_ENABLE(APX_PPR_PREVIOUS, PPR_PreviousEnable),
  EV_COMMAND_ENABLE(APX_PPR_NEXT, PPR_NextEnable),
  EV_COMMAND(APX_PPR_PREVIOUS, PPR_Previous),
  EV_COMMAND(APX_PPR_NEXT, PPR_Next),
  EV_COMMAND(APX_PPR_ONEUP, PPR_OneUp),
  EV_COMMAND_ENABLE(APX_PPR_TWOUP, PPR_TwoUpEnable),
  EV_COMMAND(APX_PPR_TWOUP, PPR_TwoUp),
  EV_COMMAND(APX_PPR_DONE, PPR_Done),
  EV_COMMAND(CM_FILEPRINT, CmPrint),
  EV_COMMAND_ENABLE(CM_FILEPRINT, CmPrintEnable),
//{{TApxPreviewWinRSP_TBL_BEGIN}}
  EV_WM_CLOSE,
//{{TApxPreviewWinRSP_TBL_END}}
END_RESPONSE_TABLE;


TApxPreviewWin::TApxPreviewWin(TWindow* parentWindow, TPrinter* printer,
                               TWindow* currWindow, const char far* title,
                               TLayoutWindow* client)
:
  TDecoratedFrame(parentWindow, title, client)
{
  CurrWindow = currWindow;
  Printer = printer;
  Client = client;
  Page1 = 0;
  Page2 = 0;
  FromPage = 1;
  ToPage = 1;

  TPrintDialog::TData& data = Printer->GetSetup();
```

```
  PrnDC = new TPrintDC(data.GetDriverName(),
                       data.GetDeviceName(),
                       data.GetOutputName(),
                       data.GetDevMode());

  PrintExtent = new TSize(PrnDC->GetDeviceCaps(HORZRES),
                          PrnDC->GetDeviceCaps(VERTRES));
  Printout = new TApxPrintout(Printer, "Print Preview", currWindow, true);

  SetBkgndColor(::GetSysColor(COLOR_APPWORKSPACE));

  // Create default toolbar New and associate toolbar buttons with commands.
  //
  PreviewSpeedBar = new TControlBar(this);
        PreviewSpeedBar->Insert(*new    TButtonGadget(APX_PPR_PREVIOUS,
APX_PPR_PREVIOUS,
                           TButtonGadget::Command, true));
  PreviewSpeedBar->Insert(*new TButtonGadget(APX_PPR_NEXT, APX_PPR_NEXT,
                           TButtonGadget::Command, true));
  PreviewSpeedBar->Insert(*new TSeparatorGadget(6));
  PreviewSpeedBar->Insert(*new TButtonGadget(APX_PPR_ONEUP, APX_PPR_ONEUP,
                                        TButtonGadget::Exclusive,  true,
TButtonGadget::Down));
  PreviewSpeedBar->Insert(*new TButtonGadget(APX_PPR_TWOUP, APX_PPR_TWOUP,
                           TButtonGadget::Exclusive, true));
  PreviewSpeedBar->Insert(*new TSeparatorGadget(12));
        PreviewSpeedBar->Insert(*new    TTextGadget(APX_PPR_CURRPAGE,
TGadget::Recessed,
                           TTextGadget::Left, 10, "Page 1"));
  PreviewSpeedBar->Insert(*new TSeparatorGadget(20));
  PreviewSpeedBar->Insert(*new TButtonGadget(CM_FILEPRINT, CM_FILEPRINT,
                           TButtonGadget::Command, true));
  PreviewSpeedBar->Insert(*new TSeparatorGadget(20));
  PreviewSpeedBar->Insert(*new TButtonGadget(APX_PPR_DONE, APX_PPR_DONE,
                           TButtonGadget::Command, true));
  Insert(*PreviewSpeedBar, TDecoratedFrame::Top);

  Attr.Style &= ~WS_VISIBLE;
  SetAcceleratorTable(IDM_PRINTPREVIEW);
}


TApxPreviewWin::~TApxPreviewWin()
{
  delete Page1;
  delete Page2;

  delete PrnDC;
  delete PrintExtent;
  delete Printout;
}


void TApxPreviewWin::SetupWindow()
{
  TDecoratedFrame::SetupWindow();

  TAPointer<char> captionText = new char[256];

  // Set the caption of the preview window based on that of the Main Window.
```

```
  //
  GetApplication()->GetMainWindow()->GetWindowText(captionText, 256);
  strcat(captionText, " (Preview)");
  SetCaption(captionText);

  // Set the icons of the preview window.
  //
  SetIcon(GetApplication(), IDI_MDIAPPLICATION);
  SetIconSm(GetApplication(), IDI_MDIAPPLICATION);

  TPrintDialog::TData& data = Printer->GetSetup();
  Page1 = new TPreviewPage(Client, *Printout, *PrnDC, *PrintExtent, 1);
  Page1->SetPageNumber(1);
  data.MaxPage = 1;

  Page2 = 0;

  TLayoutMetrics metrics1;

  metrics1.X.Set(lmLeft, lmRightOf, lmParent, lmLeft, 15);
  metrics1.Y.Set(lmTop, lmBelow, lmParent, lmTop, 15);

  // Determine major axis of preview page, have that follow parent size.
  // Make minor axis a percentage (aspect ratio) of the page's major axis
  //
  TRect r = Client->GetClientRect();
  long ratio;

  if (PrintExtent->cx > PrintExtent->cy)
    ratio = ((long)PrintExtent->cy * 100) / PrintExtent->cx;
  else
    ratio = ((long)PrintExtent->cx * 100) / PrintExtent->cy;

  bool xMajor = ((r.Width() * ratio) / 100) > r.Height();
  if (xMajor){
    metrics1.Height.Set(lmBottom, lmAbove, lmParent, lmBottom, 15);
    metrics1.Width.PercentOf(Page1,
                                          (int)((long)PrintExtent->cx * 95 /
PrintExtent->cy),
                             lmHeight);
  }
  else {
    metrics1.Height.PercentOf(Page1,
                                          (int)((long)PrintExtent->cy * 95 /
PrintExtent->cx),
                              lmWidth);
    metrics1.Width.Set(lmRight, lmLeftOf, lmParent, lmRight, 15);
  }

  Page1->Create();

  Client->SetChildLayoutMetrics(*Page1, metrics1);
  Client->Layout();
}


void TApxPreviewWin::SpeedBarState()
{
  // Update the page count.
  //
  TTextGadget* cpGadget = TYPESAFE_DOWNCAST
```

```
                              (PreviewSpeedBar->GadgetWithId(APX_PPR_CURRPAGE),
                               TTextGadget);
  if (cpGadget) {
    TAPointer<char> buffer = new char[32];

    if (Page2 && FromPage != ToPage)
      sprintf(buffer, "Page %d - %d", FromPage, ToPage);
    else
      sprintf(buffer, "Page %d", FromPage);
    cpGadget->SetText(buffer);
  }
}


void TApxPreviewWin::PPR_PreviousEnable(TCommandEnabler& tce)
{
  // Only have previous on if we're not at the first page.
  //
  tce.Enable(FromPage != 1);
}


void TApxPreviewWin::PPR_NextEnable(TCommandEnabler& tce)
{
  // Only have next on if we're not at the last page.
  //
  TPrintDialog::TData& printerData = Printer->GetSetup();
  tce.Enable(ToPage != printerData.MaxPage);
}


void TApxPreviewWin::PPR_Previous()
{
  TPrintDialog::TData& printerData = Printer->GetSetup();

  if (FromPage > printerData.MinPage) {
    FromPage--;
    ToPage--;

    Page1->SetPageNumber(FromPage);
    if (Page2)
      Page2->SetPageNumber(ToPage);
  }

  SpeedBarState();
}


void TApxPreviewWin::PPR_Next()
{
  TPrintDialog::TData& printerData = Printer->GetSetup();

  if (ToPage < printerData.MaxPage) {
    FromPage++;
    ToPage++;

    Page1->SetPageNumber(FromPage);
    if (Page2)
      Page2->SetPageNumber(ToPage);
  }
```

```
  SpeedBarState();
}


void TApxPreviewWin::PPR_OneUp()
{
  if (Page2) {
    Client->RemoveChildLayoutMetrics(*Page2);

    delete Page2;
    Page2 = 0;

    Client->Layout();

    ToPage = FromPage;

    SpeedBarState();
  }
}


void TApxPreviewWin::PPR_TwoUpEnable(TCommandEnabler& tce)
{
  // Two up is only available for portrait mode.
  //
  tce.Enable(PrintExtent->cx <= PrintExtent->cy);
}


void TApxPreviewWin::PPR_TwoUp()
{
  if (!Page2) {
    Page2 = new TPreviewPage(Client, *Printout, *PrnDC, *PrintExtent,
                             PageNumber + 1);
    Page2->Create();

    TLayoutMetrics metrics2;

    metrics2.X.Set(lmLeft, lmRightOf, Page1, lmRight, 30);
    metrics2.Y.SameAs(Page1, lmTop);

    // Assume portrait
    //
    metrics2.Width.SameAs(Page1, lmWidth);
    metrics2.Height.SameAs(Page1, lmBottom);

    Client->SetChildLayoutMetrics(*Page2, metrics2);
    Client->Layout();

    TPrintDialog::TData& printerData = Printer->GetSetup();

    // Page 2 is the next page.  If the next page is outside of our range then
    // set the first page back one and the 2nd page is the current page. If
the
    // document is only 1 page long then the 2nd page is empty.
    //
    if (FromPage == printerData.MaxPage) {
      if (FromPage > 1) {
        FromPage--;
        ToPage = FromPage + 1;
        Page1->SetPageNumber(FromPage);
```

```
        Page2->SetPageNumber(ToPage);
      }
      else
        Page2->SetPageNumber(0);
    }
    else {
      ToPage = FromPage + 1;
      Page2->SetPageNumber(ToPage);
    }

    SpeedBarState();
  }
}


void TApxPreviewWin::PPR_Done()
{
  // Don't call the base class EvClose; we do not want TApxPreviewWin to be
  // destructed.
  //
  GetApplication()->EndModal(IDCANCEL);
}


void TApxPreviewWin::EvClose()
{
  // Don't call the base class EvClose; we do not want TApxPreviewWin to be
  // destructed.
  //
  GetApplication()->EndModal(IDCANCEL);
}


void TApxPreviewWin::CmPrint()
{
  TWindow* client = GetApplication()->GetMainWindow()->GetClientWindow();

  if (client)
    client->SendMessage(WM_COMMAND, CM_FILEPRINT, 0);
}


void TApxPreviewWin::CmPrintEnable(TCommandEnabler& tce)
{
  tce.Enable(true);
}
```

```
//****************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingaboutdlg.h
//
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Class definition for GraphingAboutDlg (TDialog).
//
//  Author: Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#if !defined(graphingaboutdlg_h)    // Sentry, use file only if it's not
#define graphingaboutdlg_h          // already included.

#include <owl/static.h>

#include "graphingapp.rh"           // Definition of all resources.


//{{TDialog = GraphingAboutDlg}}
class GraphingAboutDlg : public TDialog {
  public:
    GraphingAboutDlg(TWindow* parent, TResId resId = IDD_ABOUT,
                     TModule* module = 0);
    virtual ~GraphingAboutDlg();

//{{GraphingAboutDlgVIRTUAL_BEGIN}}
  public:
    void SetupWindow();
//{{GraphingAboutDlgVIRTUAL_END}}
};    //{{GraphingAboutDlg}}


// Reading the VERSIONINFO resource.
//
class TProjectRCVersion {
  public:
    TProjectRCVersion(TModule* module);
    virtual ~TProjectRCVersion();

    bool GetProductName(LPSTR& prodName);
    bool GetProductVersion(LPSTR& prodVersion);
    bool GetCopyright(LPSTR& copyright);
    bool GetDebug(LPSTR& debug);

  protected:
    uint8 far*  TransBlock;
    void far*   FVData;

  private:
    // Don't allow this object to be copied.
    //
    TProjectRCVersion(const TProjectRCVersion&);
    TProjectRCVersion& operator = (const TProjectRCVersion&);
```

```
};

#endif  // graphingaboutdlg_h sentry.
```

```
//****************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs
//
//  SUBSYSTEM:    Graphing Application
//
//  FILE:         graphingaboutdlg.cpp
//
//  OVERVIEW
//
//  ~~~~~~~~
//
//  Source file for implementation of GraphingAboutDlg (TDialog).
//
//  Author : Alias Abdul-Rahman (c) 1999
//
//****************************************************************************//

#include <owl/pch.h>
#include <stdio.h>
#if defined(BI_PLAT_WIN16)
# include <ver.h>
#endif

#include "graphingapp.h"
#include "graphingaboutdlg.h"


TProjectRCVersion::TProjectRCVersion(TModule* module)
{
  uint32  fvHandle;
  uint    vSize;
  char    appFName[255];
  TAPointer<char> subBlockName = new char[255];

  FVData = 0;

  module->GetModuleFileName(appFName, sizeof appFName);
  OemToAnsi(appFName, appFName);
  uint32 dwSize = ::GetFileVersionInfoSize(appFName, &fvHandle);
  if (dwSize) {
    FVData  = (void far *)new char[(uint)dwSize];
    if (::GetFileVersionInfo(appFName, fvHandle, dwSize, FVData)) {
      // Copy string to buffer so if the -dc compiler switch(Put constant
      // strings in code segments)
      // is on VerQueryValue will work under Win16.  This works around
      // a problem in Microsoft's ver.dll
      // which writes to the string pointed to by subBlockName.
      //
      strcpy(subBlockName, "\\VarFileInfo\\Translation");
      if (!::VerQueryValue(FVData, subBlockName,(void far* far*)&TransBlock,
          &vSize)) {
        delete[] FVData;
        FVData = 0;
       }
      else
        // Swap the words so sprintf will print the lang-charset in the
        // correct format.
        //
        *(uint32 *)TransBlock = MAKELONG(HIWORD(*(uint32 *)TransBlock),
                                         LOWORD(*(uint32 *)TransBlock));
    }
```

```
  }
}


TProjectRCVersion::~TProjectRCVersion()
{
  if (FVData)
    delete[] FVData;
}


bool TProjectRCVersion::GetProductName(LPSTR& prodName)
{
  uint    vSize;
  TAPointer<char> subBlockName = new char[255];

  if (FVData) {
          sprintf(subBlockName,   "\\StringFileInfo\\%08lx\\%s",   *(uint32
*)TransBlock,
           (LPSTR)"ProductName");
    return FVData ? ::VerQueryValue(FVData, subBlockName,
                                    (void far* far*)&prodName, &vSize) :
false;
  } else
    return false;
}


bool TProjectRCVersion::GetProductVersion(LPSTR& prodVersion)
{
  uint    vSize;
  TAPointer<char> subBlockName = new char[255];

  if (FVData) {
          sprintf(subBlockName,   "\\StringFileInfo\\%08lx\\%s",   *(uint32
*)TransBlock,
                          (LPSTR)"ProductVersion");
    return FVData ? ::VerQueryValue(FVData, subBlockName,
                           (void far* far*)&prodVersion, &vSize) : false;
  } else
    return false;
}


bool TProjectRCVersion::GetCopyright(LPSTR& copyright)
{
  uint    vSize;
  TAPointer<char> subBlockName = new char[255];

  if (FVData) {
          sprintf(subBlockName,   "\\StringFileInfo\\%08lx\\%s",   *(uint32
*)TransBlock,
                          (LPSTR)"LegalCopyright");
    return FVData ? ::VerQueryValue(FVData, subBlockName,
                                    (void far* far*)&copyright, &vSize) :
false;
  } else
    return false;
}
```

```
bool TProjectRCVersion::GetDebug(LPSTR& debug)
{
  uint    vSize;
  TAPointer<char> subBlockName = new char[255];

  if (FVData) {
          sprintf(subBlockName,   "\\StringFileInfo\\%08lx\\%s",   *(uint32
*)TransBlock,
                            (LPSTR)"SpecialBuild");
    return FVData ? ::VerQueryValue(FVData, subBlockName,
                                    (void far* far*)&debug, &vSize) : false;
  } else
    return false;
}


//{{GraphingAboutDlg Implementation}}


//----------------------------------------------------------
// GraphingAboutDlg
// ~~~~~~~~~~~~
// Construction/Destruction handling.
//
GraphingAboutDlg::GraphingAboutDlg(TWindow* parent, TResId resId, TModule*
module)
:
  TDialog(parent, resId, module)
{
  // INSERT>> Your constructor code here.
}


GraphingAboutDlg::~GraphingAboutDlg()
{
  Destroy();

  // INSERT>> Your destructor code here.
}


void GraphingAboutDlg::SetupWindow()
{
  LPSTR prodName = 0, prodVersion = 0, copyright = 0, debug = 0;

  // Get the static text for the value based on VERSIONINFO.
  //
  TStatic* versionCtrl = new TStatic(this, IDC_VERSION, 255);
  TStatic* copyrightCtrl = new TStatic(this, IDC_COPYRIGHT, 255);
  TStatic* debugCtrl = new TStatic(this, IDC_DEBUG, 255);

  TDialog::SetupWindow();

  // Process the VERSIONINFO.
  //
  TProjectRCVersion applVersion(GetModule());

  // Get the product name and product version strings.
  //
  if (applVersion.GetProductName(prodName) &&
                          applVersion.GetProductVersion(prodVersion)) {
```

```
    // IDC_VERSION is the product name and version number, the initial value
    // of IDC_VERSION is
    // the word Version(in whatever language) product name VERSION
    // product version.
    //
    char buffer[255];
    char versionName[128];

    buffer[0] = '\0';
    versionName[0] = '\0';

    versionCtrl->GetText(versionName, sizeof versionName);
    sprintf(buffer, "%s %s %s", prodName, versionName, prodVersion);

    versionCtrl->SetText(buffer);
  }

  // Get the legal copyright string.
  //
  if (applVersion.GetCopyright(copyright))
    copyrightCtrl->SetText(copyright);

  // Only get the SpecialBuild text if the VERSIONINFO resource is there.
  //
  if (applVersion.GetDebug(debug))
    debugCtrl->SetText(debug);
}
```

```
//************************************************************************//
//  Project: User Interface (MDI) for 2D/3D TINs                          //
//                                                                        //
//  SUBSYSTEM:  Graphing Application                                      //
//  FILE:     graphingapp.rc                                              //
//  AUTHOR:                                                               //
//                                                                        //
//  OVERVIEW                                                              //
//  ~~~~~~~~                                                              //
//  All resources defined here.                                           //
//                                                                        //
//  Author: Alias Abdul-Rahman (c) 1999                                   //
//************************************************************************//

#if !defined(WORKSHOP_INVOKED)
# include <windows.h>
#endif
#include "graphingapp.rh"


IDM_MDI MENU
BEGIN
  POPUP "&File"
  BEGIN
    MENUITEM "&New", CM_MDIFILENEW
    MENUITEM "&Open Input Files...", CM_MDIFILEOPEN
    MENUITEM "&Close", CM_FILECLOSE
    MENUITEM SEPARATOR
    MENUITEM "&Save", CM_FILESAVE, GRAYED
    MENUITEM "Save &As...", CM_FILESAVEAS, GRAYED
    MENUITEM SEPARATOR
    MENUITEM "Print Pre&view...", CM_FILEPRINTPREVIEW, GRAYED
    MENUITEM "&Print...", CM_FILEPRINT, GRAYED
    MENUITEM "P&rint Setup...", CM_FILEPRINTERSETUP, GRAYED
    MENUITEM SEPARATOR
    MENUITEM "E&xit\tAlt+F4", CM_EXIT
  END

  MENUITEM SEPARATOR

  //POPUP "&Edit"
  //BEGIN
    //MENUITEM "&Undo\tAlt+BkSp", CM_EDITUNDO, GRAYED
    //MENUITEM SEPARATOR
    //MENUITEM "Cu&t\tShift+Del", CM_EDITCUT, GRAYED
    //MENUITEM "&Copy\tCtrl+Ins", CM_EDITCOPY, GRAYED
    //MENUITEM "&Paste\tShift+Ins", CM_EDITPASTE, GRAYED
    //MENUITEM SEPARATOR
    //MENUITEM "Clear &All\tCtrl+Del", CM_EDITCLEAR, GRAYED
    //MENUITEM "&Delete\tDel", CM_EDITDELETE, GRAYED
  //END

  //POPUP "&Search"
  //BEGIN
    //MENUITEM "&Find...", CM_EDITFIND, GRAYED
    //MENUITEM "&Replace...", CM_EDITREPLACE, GRAYED
    //MENUITEM "&Next\aF3", CM_EDITFINDNEXT, GRAYED
  //END

  MENUITEM SEPARATOR
```

```
  MENUITEM SEPARATOR

  MENUITEM SEPARATOR

  POPUP "&Window"
  BEGIN
    MENUITEM "&Cascade", CM_CASCADECHILDREN
    MENUITEM "&Tile", CM_TILECHILDREN
    MENUITEM "Arrange &Icons", CM_ARRANGEICONS
    MENUITEM "C&lose All", CM_CLOSECHILDREN
  END

  MENUITEM SEPARATOR

  POPUP "&Help"
  BEGIN
    MENUITEM "&About...", CM_HELPABOUT
  END

END


// Accelerator table for short-cut to menu commands. (include/owl/editfile.rc)
//
//IDM_MDI ACCELERATORS
//BEGIN
  //VK_DELETE, CM_EDITCUT, VIRTKEY, SHIFT
  //VK_INSERT, CM_EDITCOPY, VIRTKEY, CONTROL
  //VK_INSERT, CM_EDITPASTE, VIRTKEY, SHIFT
  //VK_DELETE, CM_EDITCLEAR, VIRTKEY, CONTROL
  //VK_BACK,   CM_EDITUNDO, VIRTKEY, ALT
  //VK_F3,     CM_EDITFINDNEXT, VIRTKEY
//END


// Accelerator table for Print Preview window.
//
IDM_PRINTPREVIEW ACCELERATORS
BEGIN
  VK_ESCAPE, APX_PPR_DONE, VIRTKEY
  "c",       APX_PPR_DONE, ALT
END



// Menu merged in when TEditView is active, notice the extra MENUITEM
SEPARATORs which are
// for menu negotation.  These separators are used as group markers by OWL.
//
IDM_EDITVIEW MENU LOADONCALL MOVEABLE PURE DISCARDABLE
BEGIN
  MENUITEM SEPARATOR

  //POPUP "&Edit"
  //BEGIN
    //MENUITEM "&Undo\aCtrl+Z", CM_EDITUNDO
    //MENUITEM SEPARATOR
    //MENUITEM "&Cut\aCtrl+X", CM_EDITCUT
    //MENUITEM "C&opy\aCtrl+C", CM_EDITCOPY
    //MENUITEM "&Paste\aCtrl+V", CM_EDITPASTE
    //MENUITEM "&Delete\aDel", CM_EDITDELETE
    //MENUITEM "C&lear All\aCtrl+Del", CM_EDITCLEAR
```

```
  //END

  //POPUP "&Search"
  //BEGIN
    //MENUITEM "&Find...", CM_EDITFIND
    //MENUITEM "&Replace...", CM_EDITREPLACE
    //MENUITEM "&Next\aF3", CM_EDITFINDNEXT
  //END

  MENUITEM SEPARATOR

  MENUITEM SEPARATOR

  MENUITEM SEPARATOR

  MENUITEM SEPARATOR
END


// Menu merged in when TListView is active, notice the extra MENUITEM
SEPARATORs which are
// for menu negotation.  These separators are used as group markers by OWL.
//
IDM_LISTVIEW MENU LOADONCALL MOVEABLE PURE DISCARDABLE
BEGIN
  MENUITEM SEPARATOR

  //POPUP "&Edit"
  //BEGIN
    //MENUITEM "&Undo\aCtrl+Z", CM_EDITUNDO
    //MENUITEM SEPARATOR
    //MENUITEM "&Cut\aCtrl+X", CM_EDITCUT
    //MENUITEM "C&opy\aCtrl+C", CM_EDITCOPY
    //MENUITEM "&Paste\aCtrl+V", CM_EDITPASTE
    //MENUITEM "&Delete\aDel", CM_EDITDELETE
    //MENUITEM "&Add Item\aIns", CM_EDITADD
    //MENUITEM "&Edit Item\aEnter", CM_EDITEDIT
    //MENUITEM "C&lear All\aCtrl+Del", CM_EDITCLEAR
  //END

  MENUITEM SEPARATOR

  MENUITEM SEPARATOR

  MENUITEM SEPARATOR

  MENUITEM SEPARATOR
END


IDM_DOCMANAGERFILE MENU LOADONCALL MOVEABLE PURE DISCARDABLE
BEGIN
  MENUITEM "&New", CM_MDIFILENEW
  MENUITEM "&Open...", CM_MDIFILEOPEN
  MENUITEM "&Close", CM_FILECLOSE
  MENUITEM SEPARATOR
  MENUITEM "&Save", CM_FILESAVE, GRAYED
  MENUITEM "Save &As...", CM_FILESAVEAS, GRAYED
  MENUITEM SEPARATOR
  MENUITEM "Print Pre&view...", CM_FILEPRINTPREVIEW, GRAYED
  MENUITEM "&Print...", CM_FILEPRINT, GRAYED
  MENUITEM "P&rint Setup...", CM_FILEPRINTERSETUP, GRAYED
```

```
  MENUITEM SEPARATOR
  MENUITEM "E&xit\tAlt+F4", CM_EXIT
END


// Table of help hints displayed in the status bar.
//
STRINGTABLE
BEGIN
  -1,                   "File/document operations"
  CM_MDIFILENEW,        "Creates a new document"
  CM_MDIFILEOPEN,       "Opens an existing document"
  CM_VIEWCREATE,        "Creates a new view for this document"
  CM_FILEREVERT,        "Reverts changes to last document save"
  CM_FILECLOSE,         "Closes the active document"
  CM_FILESAVE,          "Saves the active document"
  CM_FILESAVEAS,        "Saves the active document with a new name"
  CM_FILEPRINT,         "Prints the active document"
  CM_FILEPRINTERSETUP,  "Sets print characteristics for the active document"
  CM_FILEPRINTPREVIEW,  "Displays full pages as read-only"
  CM_EXIT,              "Quits Graphing and prompts to save the documents"
  //CM_EDITUNDO-1,        "Edit operations"
  //CM_EDITUNDO,          "Reverses the last operation"
  //CM_EDITCUT,           "Cuts the selection and puts it on the Clipboard"
  //CM_EDITCOPY,          "Copies the selection and puts it on the Clipboard"
  //CM_EDITPASTE,         "Inserts the Clipboard contents at the insertion
point"
  //CM_EDITDELETE,        "Deletes the selection"
  //CM_EDITCLEAR,         "Clears the active document"
  //CM_EDITADD,           "Inserts a new line"
  //CM_EDITEDIT,          "Edits the current line"
  //CM_EDITFIND-1,        "Search/replace operations"
  //CM_EDITFIND,          "Finds the specified text"
  //CM_EDITREPLACE,       "Finds the specified text and changes it"
  //CM_EDITFINDNEXT,      "Finds the next match"
  CM_CASCADECHILDREN-1, "Window arrangement and selection"
  CM_CASCADECHILDREN,   "Cascades open windows"
  CM_TILECHILDREN,      "Tiles open windows"
  CM_ARRANGEICONS,      "Arranges iconic windows along bottom"
  CM_CLOSECHILDREN,     "Closes all open windows"
  CM_HELPABOUT-1,       "Access About"
  CM_HELPABOUT,         "About the Graphing application"
END


//
// OWL string table
//

// EditFile (include/owl/editfile.rc and include/owl/editsear.rc)
//
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
  IDS_CANNOTFIND,       "Cannot find ""%s""."
  IDS_UNABLEREAD,       "Unable to read file %s from disk."
  IDS_UNABLEWRITE,      "Unable to write file %s to disk."
  IDS_FILECHANGED,      "The text in the %s file has changed.\n\nDo you want
to save the changes?"
  IDS_FILEFILTER,       "Text files|*.txt|AllFiles|*.*|"
END
```

```
// ListView (include/owl/listview.rc)
//
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
  IDS_LISTNUM,   "Line number %d"
END


// Doc/View (include/owl/docview.rc)
//
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
  IDS_DOCMANAGERFILE,    "&File"
  IDS_DOCLIST,           "--Document Type--"
  IDS_VIEWLIST,          "--View Type--"
  IDS_UNTITLED,          "Document"
  IDS_UNABLEOPEN,        "Unable to open document."
  IDS_UNABLECLOSE,       "Unable to close document."
  IDS_READERROR,         "Document read error."
  IDS_WRITEERROR,        "Document write error."
  IDS_DOCCHANGED,       "The document has been changed.\n\nDo you want to save
the changes?"
  IDS_NOTCHANGED,        "The document has not been changed."
  IDS_NODOCMANAGER,      "Document Manager not present."
  IDS_NOMEMORYFORVIEW,   "Insufficient memory for view."
  IDS_DUPLICATEDOC,      "Document already loaded."
END


// Printer (include/owl/printer.rc)
//
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
  IDS_PRNON,             " on "
  IDS_PRNERRORTEMPLATE, "'%s' not printed. %s."
  IDS_PRNOUTOFMEMORY,    "Out of memory"
  IDS_PRNOUTOFDISK,      "Out of disk space"
  IDS_PRNCANCEL,         "Printing canceled"
  IDS_PRNMGRABORT,       "Printing aborted in Print Manager"
  IDS_PRNGENERROR,       "Error encountered during print"
  IDS_PRNERRORCAPTION,   "Print Error"
END


// Exception string resources (include/owl/except.rc)
//
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
  IDS_OWLEXCEPTION,      "ObjectWindows Exception"
  IDS_UNHANDLEDXMSG,     "Unhandled Exception"
  IDS_OKTORESUME,        "OK to resume?"
  IDS_UNKNOWNEXCEPTION, "Unknown exception"

  IDS_UNKNOWNERROR,      "Unknown error"
  IDS_NOAPP,             "No application object"
  IDS_OUTOFMEMORY,       "Out of memory"
  IDS_INVALIDMODULE,     "Invalid module specified for window"
  IDS_INVALIDMAINWINDOW,"Invalid MainWindow"
  IDS_VBXLIBRARYFAIL,    "VBX Library init failure"

  IDS_INVALIDWINDOW,     "Invalid window %s"
  IDS_INVALIDCHILDWINDOW,"Invalid child window %s"
```

```
  IDS_INVALIDCLIENTWINDOW,"Invalid client window %s"

  IDS_CLASSREGISTERFAIL,"Class registration fail for window %s"
  IDS_CHILDREGISTERFAIL,"Child class registration fail for window %s"
  IDS_WINDOWCREATEFAIL, "Create fail for window %s"
  IDS_WINDOWEXECUTEFAIL,"Execute fail for window %s"
  IDS_CHILDCREATEFAIL,  "Child create fail for window %s"

  IDS_MENUFAILURE,      "Menu creation failure"
  IDS_VALIDATORSYNTAX,  "Validator syntax error"
  IDS_PRINTERERROR,     "Printer error"

 IDS_LAYOUTINCOMPLETE, "Incomplete layout constraints specified in window %s"
   IDS_LAYOUTBADRELWIN,    "Invalid relative window specified in layout
constraint in window %s"

  IDS_GDIFAILURE,       "GDI failure"
  IDS_GDIALLOCFAIL,     "GDI allocate failure"
  IDS_GDICREATEFAIL,    "GDI creation failure"
  IDS_GDIRESLOADFAIL,   "GDI resource load failure"
  IDS_GDIFILEREADFAIL,  "GDI file read failure"
  IDS_GDIDELETEFAIL,    "GDI object %X delete failure"
  IDS_GDIDESTROYFAIL,   "GDI object %X destroy failure"
  IDS_INVALIDDIBHANDLE, "Invalid DIB handle %X"
END


// General Window's status bar messages. (include/owl/statusba.rc)
//
STRINGTABLE
BEGIN
  IDS_MODES             "EXT|CAPS|NUM|SCRL|OVR|REC"
  IDS_MODESOFF          "   |   |   |   |   |   "
  SC_SIZE,              "Changes the size of the window"
  SC_MOVE,              "Moves the window to another position"
  SC_MINIMIZE,          "Reduces the window to an icon"
  SC_MAXIMIZE,          "Enlarges the window to it maximum size"
  SC_RESTORE,           "Restores the window to its previous size"
  SC_CLOSE,             "Closes the window"
  SC_TASKLIST,          "Opens task list"
  SC_NEXTWINDOW,        "Switches to next window"
END


// Validator messages (include/owl/validate.rc)
//
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
  IDS_VALPXPCONFORM     "Input does not conform to picture:\n""%s"""
  IDS_VALINVALIDCHAR    "Invalid character in input"
  IDS_VALNOTINRANGE     "Value is not in the range %ld to %ld."
  IDS_VALNOTINLIST      "Input is not in valid-list"
END


//
// Bitmaps used by the speedbar.  Each bitmap is associated with a
// particular menu command.
//
CM_MDIFILENEW BITMAP "new.bmp"
CM_MDIFILEOPEN BITMAP "pointfile.bmp"
CM_FILESAVE BITMAP "save.bmp"
```

```
//CM_EDITUNDO BITMAP "undo.bmp"
//CM_EDITCUT BITMAP "cut.bmp"
//CM_EDITCOPY BITMAP "copy.bmp"
//CM_EDITPASTE BITMAP "paste.bmp"

//CM_EDITFIND BITMAP "find.bmp"
//CM_EDITFINDNEXT BITMAP "findnext.bmp"

CM_FILEPRINTPREVIEW  BITMAP "preview.bmp"

CM_FILEPRINT BITMAP "print.bmp"


//
// Print Preview speed bar bitmaps
//
APX_PPR_PREVIOUS BITMAP "previous.bmp"
APX_PPR_NEXT BITMAP "next.bmp"
APX_PPR_ONEUP BITMAP "preview1.bmp"
APX_PPR_TWOUP BITMAP "preview2.bmp"
APX_PPR_DONE BITMAP "prexit.bmp"


//
// Misc application definitions
//

// MDI document ICON
//
IDI_DOC ICON "mdichild.ico"

// Application ICON
//
IDI_MDIAPPLICATION ICON "appldocv.ico"


// About box.
//
IDD_ABOUT DIALOG 12, 17, 204, 65
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About 2D/3D TINs"
FONT 8, "MS Sans Serif"
BEGIN
  CTEXT "Version", IDC_VERSION, 2, 14, 200, 8, SS_NOPREFIX
  CTEXT "The 2D/3D TINs program", -1, 2, 4, 200, 8, SS_NOPREFIX
  CTEXT "", IDC_COPYRIGHT, 2, 27, 200, 17, SS_NOPREFIX
  RTEXT "", IDC_DEBUG, 136, 55, 66, 8, SS_NOPREFIX
  ICON IDI_MDIAPPLICATION, -1, 2, 2, 34, 34
  DEFPUSHBUTTON "OK", IDOK, 82, 48, 40, 14
END


// Printer abort box.
//
IDD_ABORTDIALOG DIALOG 84, 51, 130, 60
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Printing"
BEGIN
  PUSHBUTTON "Cancel", IDCANCEL, 46, 40, 40, 14, WS_TABSTOP
   CTEXT "Now printing Page %d of", ID_PAGE, 0, 8, 130, 8, SS_CENTER | NOT
WS_VISIBLE | WS_GROUP
  CTEXT "Now printing", -1, 0, 8, 130, 8,
```

```
  CTEXT "'%s' on the", ID_TITLE, 0, 16, 130, 8
  CTEXT "", ID_PORT, 0, 24, 130, 8, SS_CENTER | NOT WS_VISIBLE | WS_GROUP
  CTEXT "%s on %s", ID_DEVICE, 0, 24, 130, 8
END


// Version info.
//
#if !defined(__DEBUG_)

// Non-Debug VERSIONINFO
//
1 VERSIONINFO LOADONCALL MOVEABLE
FILEVERSION 1, 0, 0, 0
PRODUCTVERSION 1, 0, 0, 0
FILEFLAGSMASK 0
FILEFLAGS VS_FFI_FILEFLAGSMASK
#if defined(BI_PLAT_WIN32)
FILEOS VOS__WINDOWS32
#else
FILEOS VOS__WINDOWS16
#endif
FILETYPE VFT_APP
BEGIN
  BLOCK "StringFileInfo"
  BEGIN
     // Language type = U.S. English(0x0409) and Character Set = Windows,
Multilingual(0x04e4)
     BLOCK "040904E4"                // Matches VarFileInfo Translation hex
value.
    BEGIN
      VALUE "CompanyName", "\000"
      VALUE "FileDescription", "2D TIN & 3D TIN for Windows\000"
      VALUE "FileVersion", "1.0\000"
      VALUE "InternalName", "TIN\000"
      VALUE "LegalCopyright", "Copyright © 1999. All Rights Reserved.\000"
        VALUE "LegalTrademarks", "Windows(TM) is a trademark of Microsoft
Corporation\000"
      VALUE "OriginalFilename", "Graphing.exe\000"
      VALUE "ProductName", "2D/3D TIN\000"
      VALUE "ProductVersion", "1.0\000"
    END
  END

  BLOCK "VarFileInfo"
  BEGIN
    VALUE "Translation", 0x0409, 0x04e4    // U.S. English(0x0409) & Windows
Multilingual(0x04e4) 1252
  END
END

#else

// Debug VERSIONINFO
//
1 VERSIONINFO LOADONCALL MOVEABLE
FILEVERSION 1, 0, 0, 0
PRODUCTVERSION 1, 0, 0, 0
FILEFLAGSMASK   VS_FF_DEBUG   |   VS_FF_PRERELEASE   |   VS_FF_PATCHED   |
VS_FF_PRIVATEBUILD | VS_FF_SPECIALBUILD
FILEFLAGS VS_FFI_FILEFLAGSMASK
#if defined(BI_PLAT_WIN32)
```

```
FILEOS VOS__WINDOWS32
#else
FILEOS VOS__WINDOWS16
#endif
FILETYPE VFT_APP
BEGIN
  BLOCK "StringFileInfo"
  BEGIN
     // Language type = U.S. English(0x0409) and Character Set = Windows,
Multilingual(0x04e4)
    BLOCK "040904E4"                // Matches VarFileInfo Translation hex
value.
    BEGIN
      VALUE "CompanyName", "\000"
      VALUE "FileDescription", "2D/3D TIN for Windows\000"
      VALUE "FileVersion", "1.0\000"
      VALUE "InternalName", "2D/3D TIN\000"
      VALUE "LegalCopyright", "Copyright © 1999. All Rights Reserved.\000"
        VALUE "LegalTrademarks", "Windows(TM) is a trademark of Microsoft
Corporation\000"
      VALUE "OriginalFilename", "Graphing.exe\000"
      VALUE "ProductName", "2D/3D TIN\000"
      VALUE "ProductVersion", "1.0\000"
      VALUE "SpecialBuild", "Debug Version\000"
      VALUE "PrivateBuild", "Built by Alias Abdul-Rahman \000"
    END
  END

  BLOCK "VarFileInfo"
  BEGIN
    VALUE "Translation", 0x0409, 0x04e4    // U.S. English(0x0409) & Windows
Multilingual(0x04e4) 1252
  END
END

#endif
```